University of Zagreb

FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS

Ana Klobučar

# Solving robust variants of the maximum weighted independent set problem

DOCTORAL DISSERTATION

Zagreb, 2019.

University of Zagreb

FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS

Ana Klobučar

# Solving robust variants of the maximum weighted independent set problem

DOCTORAL DISSERTATION

Supervisor:

professor Robert Manger

Zagreb, 2019.

Sveučilište u Zagrebu

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Klobučar

# Rješavanje robusnih varijanti problema maksimalnog težinskog nezavisnog skupa

DOKTORSKI RAD

Mentor:

Prof. dr. sc. Robert Manger

Zagreb, 2019.

# About mentor

Personal information

- Name and surname: Robert Manger
- Place and date of birth: Zagreb, 15/04/1957
- Scientific Personal Identification Number: 93880
- Link to the CROSBI profile: http://bib.irb.hr/lista-radova?autor=93880
- Web page: http://web.math.pmf.unizg.hr/ manger/

Education

- 1990: PhD, University of Zagreb, Faculty of Science, Department of Mathematics.
- 1982: MSc, University of Zagreb, Faculty of Science, Department of Mathematics.
- 1979: BSc, University of Zagreb, Faculty of Science, Department of Mathematics.

Present employment

- 2007- today: Full professor. University of Zagreb, Faculty of Science, Department of Mathematics.

Work on projects

- 2015-2016: collaborator on a project EkoRaMa financed by the European Union.
- 1997-2013: principal investigator for three research projects financed by the Croatian Ministry of Science, Education and Sports.
- 2004-2006: collaborator on a technological development project financed by the Croatian Ministry of Science, Education and Sports.

- 1993-1996: collaborator on three research projects financed by the Croatian Ministry of Science and Technology.

Publications

- 1991-2015: 25 research papers published in international scientific journals.
- 1987-2008: 25 research papers published in international scientific conference proceedings.
- 1994: 2 research papers published as chapters in a book.
- 2012-2016: 3 textbooks approved by the University of Zagreb.
- 1980-1998: 5 professional papers published in journals.
- 1982-2006: 7 professional papers published in conference proceedings.
- 2001, 2008: 2 professional papers published in books.

Mentorships

- 1993-2017: cca 140 “diploma” works
- 1997-2000: 4 master theses
- 1999-2011: 5 doctoral dissertations.

# ACKNOWLEDGEMENTS

To my dear parents Antoaneta and Davor Klobučar, helpful mentor Robert Manger and loving husband Tomislav Barišić. Thank you for your for support in writing this thesis and in my life in general.

# SUMMARY

This work is concerned with robust variants of the maximum weighted independent set problem (MWIS problem). Three basic robustness criteria are used, i.e. absolute robustness, robust deviation and relative robust deviation. More general ordered weighted averaging criteria (OWA) are also considered. Problems are posed in a graph whose vertices are given weights. Uncertainty in vertex weights is expressed through a finite collection of explicitly given scenarios or by intervals. First we explore relationship between robust variants of our problem and robust variants of minimum weighted vertex cover problem (MWVC). In more detail, we explore whether the complement of a robustly optimal independent set must be a robustly optimal vertex cover, and vice-versa (as it is true for conventional optima). It turns out that the answer to this question is not straightforward. More precisely, the answer depends on the chosen criterion of robustness.

Further, since solving the conventional MWIS problem is already NP-hard, finding the exact solution of its robust counterpart obviously cannot be any easier. Therefore, we propose an approximate algorithm for solving the considered robust variants, which is based on evolutionary computing and on various crossover and mutation operators. The algorithm is experimentally evaluated on appropriate problem instances. It is shown that satisfactory solutions can be obtained for the mentioned robust robustness criteria in reasonable time.

Finally, we explore complexity of our robust variants on trees. It is well known that the conventional MWIS problem can be solved in polynomial time on trees. However, it turns out that almost all robust variants are NP-hard. Hence, we propose an approximative algorithm specially designed for trees which takes into consideration their special structure. More precisely, the algorithm combines good features from dynamic programming, evolutionary computing and greedy decision making. Again, the algorithm is experimentally evaluated on appropriate problem instances. It is shown that satisfactory solutions can be obtained for any of the three basic robustness criteria in reasonable time.

Key word: robust optimization, maximum weighted independent set, complexity, approximative algorithms, minimum weighted vertex cover, graphs, trees

# SAŽETAK (PROŠIRENI)

Konvencionalna optimizacija podrazumijeva maksimiziranje ili minimiziranje funkcije cilja nad danim skupom dopustivih rješenja koja zadovoljavaju određena ograničenja. Međutim, u stvarnim situacijama, ulazni podaci su često nepoznati i podložni promjenama. Suvremena metoda za rad s takvim nesigurnostima se zove robusna optimizacija. Ovaj rad se bavi robusnim varijantama problema maksimalnog težinskog nezavisnog skupa (problem MTNS). Koriste se tri osnovna kriterija robusnosti: apsolutna robusnost, robusna devijacija te relativna robusna devijacija. Također se promatraju i općenitiji OWA kriteriji. Nesigurnost u pogledu težina vrhova izražena je preko eksplicitnog skupa scenarija ili pomoću intervala.

Neka je $G=(V,E)$ neusmjereni graf, gdje je $V$ skup vrhova, a $E$ skup bridova. *Nezavisni skup* od $G$ je podskup skupa vrhova $X \subseteq V$ takav da ne postoje dva vrha iz $X$ koja su susjedna (povezana bridom iz $E$). Nadalje, neka je $G=(V,E)$ neusmjereni graf čiji vrhovi imaju cjelobrojne nenegativne težine. *Maksimalni težinski nezavisni skup* od $G$ je nezavisni skup od $G$ takav da je zbroj težina vrhova najveći mogući. Problem nalaženja takvog skupa za dani graf se naziva *(konvencionalni) problem maksimalnog težinskog nezavisnog skupa (MTNS)*.

Da bismo smo definirali robusne varijante problema tj. njihova rješenja, prvo uvodimo sljedeće oznake. Označimo sa $S$ skup svih scenarija i pretpostavimo da je svaki scenarij zadan s $n$-torkom uređenih brojeva gdje $n$ predstavlja broj vrhova. Za $X$ proizvoljni nezavisni skup, $F(X,s)$ je vrijednost funkcije cilja konvencionalnog problema za scenarij $s$. Funkcija $F(X,s)$ je jednaka zbroju težina vrhova iz skupa $X$ za scenarij $s$. Nadalje $F_s^*$ je optimalna vrijednost funkcije cilja konvencionalnog problema za scenarij $s$. $\Phi$ je kolekcija svih nezavisnih skupova. *Apsolutno robusno rješenje $X_A$* je nezavisni skup čiji je minimum funkcije cilja, mjerene po svim scenarijima, najveći mogući t.j.

$$\mathrm{opt}_A = \max_{X\in\Phi}\min_{s\in S} F(X,s) = \max_{X\in\Phi} F_A(X) = F_A(X_A).$$

*Robusno devijantno rješenje $X_D$* je nezavisni skup čije je maksimalno odstupanje od konven-

cionalnog optimuma, mjerenog po svim scenarijima, najmanje moguće t.j.

$$\mathrm{opt}_D = \min_{X\in\Phi} \max_{s\in S} \left(F_s^* - F(X,s)\right) = \min_{X\in\Phi} F_D(X) = F_D(X_D).$$

*Releativno robusno devijantno rješenje $X_R$* je nezavisni skup čije je maksimalno relativno odstupanje od konvencionalnog optimuma, mjerenog po svim scenarijima, najmanje moguće t.j.

$$\mathrm{opt}_R = \min_{X\in\Phi} \max_{s\in S} \left((F_s^* - F(X,s))/F_s^*\right) = \min_{X\in\Phi} F_R(X) = F_R(X_R).$$

OWA kriteriji su generalizacija navedenih kriterija. Primjerice, OWA$_A$ se definira na sljedeći način. Za odabrano rješenje $X$ pripadne vrijednosti funkcije cilja $F(X,s)$ sortiramo uzlazno:

$$F(X,s_{\sigma(1)}) \leq F(X,s_{\sigma(2)}) \leq \ldots \leq F(X,s_{\sigma(p)}),$$

$\sigma$ je peremutacija. Tada se $OWA_A$ cijena za $X$ računa:

$$O_A(X) = \sum_{i=1}^{p} a_i \cdot F(X,s_{\sigma(i)}).$$

*$OWA_A$ rješenje $X_{OA}$* je nezavisni skup koje maksimizira funkciju $O_A(X)$ na skupu svih mogućih rješenja t.j.

$$\mathrm{opt}_{OA} = O_A(X_{OA}) = \max_{X\in\Phi} O_A(X).$$

U radu prvo istražujemo odnos između robusnih varijanti našeg problema i robusnih varijanti problema minimalnog težinskog vršnog pokrivača (problem MTVP). Definirajmo problem MTVP. Neka je $G = (V,E)$ neusmjereni graf, gdje je $V$ skup vrhova, a $E$ skup bridova. *Vršni pokrivač* od $G$ je podskup skupa vrhova $Y \subseteq V$ takav da je svaki brid iz $E$ incidentan s barem jednim vrhom iz $Y$. Neka je $G = (V,E)$ neusmjereni graf čiji vrhovi imaju cjelobrojne nenegativne težine. *Minimalni težinski vršni pokrivač* od $G$ je vršni pokrivač od $G$ takav da je zbroj težina vrhova najmanji mogući. Problem nalaženja takvog skupa za dani graf se naziva *(konvencionalni) problem minimalnog težinskog vršnog pokrivača (MTVP)*.

Istražujemo je li komplement robusno optimalnog nezavisnog skupa robusno optimalni vršni pokrivač i obratno (kao što vrijedi za konvencionalne optimume). Pokazuje se da odgovor na to pitanje nije trivijalan. Točnije, odgovor ovisi o odabranom kriteriju robusnosti. Komplement robusnog devijantnog rješenja problema MTNS je robusno devijantno rješenje problema MTVP i obratno. Za ostale kriterije ne vrijedi ekvivalencija osim u nekim specijalnim slučajevima.

Potom se bavimo rješavanjem robusnih varijanti problema MTNS na običnim grafovima. Kako je rješavanje konvencionalnog problema MTNS već NP-teško, nalaženje egzaktnog rješenja

njegove robusne varijante neće biti ništa lakše. Stoga predlažemo približni algoritam za rješavanje spomenutih robusnih varijanti koji se temelji na evolucijskom računanju i na kolekciji različitih operatora križanja i mutacija. Navedeni algoritam je eksperimentalno testiran na odgovarajućim instancama problema. Pokazuje se da je moguće postići zadovoljavajuća rješenja u prihvatljivom vremenu.

Konačno, istražujemo složenost navedenih robusnih varijanti na stablima. Poznato je da je konvencionalni problem MTNS na stablima rješiv u polinomijalnom vremenu. Nažalost, pokazuje se da su gotovo sve robusne varijante NP teške. Stoga predlažemo približni algoritam dizajniran specijalno za stabla koji uzima u obzir njihovu specifičnu strukturu. Detaljnije, algoritam kombinira dobre osobine dinamičkog programiranja, evolucijskog računanja i pohlepnog odlučivanja. Navedeni algoritam je također eksperimentalno testiran na odgovarajućim instancama problema. Opet se pokazuje da je moguće postići zadovoljavajuća rješenja u prihvatljivom vremenu.

Ključne riječi: robusna optimizacija, maksimalni težinski nezavisni skup, složenost, približni algoritmi, minimalni težinski vršni pokrivač, grafovi, stabla

# Contents

# INTRODUCTION

A conventional optimization problem consists of maximizing or minimizing an objective function over a set of feasible solutions that satisfy given constraints. However, in real-life situations, input parameters that specify a particular problem instance are often uncertain or subject to change, since they may be influenced by some unpredictable future circumstances. We can try to somehow estimate or approximate those parameters, but such estimations could easily lead to an inferior or even infeasible solution. Instead of ignoring uncertainty, it is much better to handle it by using an appropriate mathematical model.

A state-of-the art approach to deal with the mentioned uncertainty is called robust optimization [2, 19]. According to that approach, a finite or infinite set of scenarios is defined. The scenarios should capture uncertainty in problem parameters in some way. Only those solutions that are feasible for all scenarios are considered. As the optimal solution in the robust sense, the one is chosen whose worst behavior over the whole set of scenarios happens to be the best among all solutions. Such solution does not need to be really optimal in the conventional sense, but it is chosen in order to be acceptable even in the most adverse circumstances.

In order to have a better flexibility in modelling uncertainty and to find compromise between over-pessimism for some scenarios and over-optimism for the others, we can also use a convex combination of different behavior measurements for the same solution under different scenarios. The resulting criteria, usually called OWA, can be regarded as generalizations of the previously mentioned basic ”best-worst” criteria.

In this work we consider the maximum weighted independent set problem, which is posed in a graph whose vertices are given weights. In its conventional variant, the problem consists of finding a subset of graph vertices that are not adjacent to each other and whose sum of weights is as large as possible. Our focus is on several robust variants of the same problem, where the graph structure is fixed but vertex weights are uncertain. Such uncertainty is expressed by a finite set of scenarios.

The considered problem has many applications, e.g. in resource allocation or in different counting problems such as counting exact covers or exact hitting sets. It is obvious that in the case of resource allocation problem parameters are almost always uncertain, so that robust problem variants should be applied. Indeed, parameters dealing with placement of factories or warehouses depend heavily on market circumstances, which are volatile and hard to predict. Further, it is well known [12] that the conventional maximum weighted independent set problem is already NP-hard. Thus finding the exact solutions of its robust variants should be even harder.

Chapter 1 lists all necessary definitions and preliminaries such as robust optimization, robust variants of problems, independent sets, vertex covers etc.

In this work we first explore relationships among robust variants of the mentioned problem and robust variants of the minimum weighted vertex cover problem. The minimum weighted vertex cover problem consists of finding such a subset of graph vertices that every edge is incident on at least one vertex from the subset and whose sum of weights is as small as possible. It is well known that the complement of a maximum weighted independent set is a minimum weighted vertex cover, and vice-versa. Therefore an algorithm for finding one type of optimal solution can be used to solve the other type. But such equivalence of optimization problems is granted only within the context of conventional optimization. We cannot be sure that analogue properties will hold when we move to robust optimization. Chapter 2 gives an answer to a natural question: is the same equivalence also true if we consider robust variants of the same problems?

Although the exact algorithms may be good in theory, we do not expect that real-life problem instances could be solved to optimality in reasonable time. Instead, we believe that the only practical way of solving such instances is by using approximate algorithms and heuristics. Chapter 3 proposes an approximate algorithm for solving the different robust variants, which is based on evolutionary computing and on various crossover and mutation operators.

Finally, Chapter 4 deals with maximum weighted independent sets in trees. As the conventional maximum weighted independent set problem is solvable on trees in linear time, we expect that its robust variants may also be solved on trees more efficiently than on general graphs. Trees are interesting to study because they generalize linear graphs. Conveyor belt work can be modeled by linear graphs. Workers or machines can not access the same resources at the same time. Hence, in Chapter 4 we give the computational complexity for some robust variants on trees and an approximative algorithm specially designed for trees. The mentioned algorithm is inspired by the algorithm for the conventional problem [8].

# 1. Definitions and preliminaries

In this chapter we review all definitions and preliminary results that are necessary for the remaining chapters. First we define robust variants of optimization problems according to three basic criteria of robustness or their OWA extensions. Next we present an alternative way of defining robustness based on Pareto-efficient solutions. Also we briefly discuss how uncertainty within a robust problem can be expressed by intervals rather than with discrete scenarios. Finally, we define the conventional variant of the maximum weighted independent set problem, as well as the conventional variant of the closely related minimum weighted vertex cover problem.

## 1.1. Criteria of robustness

The foundations of robust optimization have been laid out in the seminal works [3–5, 19]. More recent surveys and some general results can be found in [1, 2, 6, 17]. Among the above references, the most important for our purposes is the book [19], which provides a framework for robust *discrete* optimization.

According to [19], uncertainty in problem parameters should be captured by a finite and explicitly given set of *scenarios*. Each scenario specifies a possible combination of parameter values. As mentioned before, only those solutions that are feasible for all scenarios are considered. The behavior of any considered solution under any scenario is measured according to some *criterion of robustness*. Then, the so-called *robustly optimal* solution is chosen as the one whose worst behavior, measured over all scenarios, is the best possible.

The framework from [19] obviously allows many variants. For instance, the set of uncertain parameters can be more or less extensive. Also, the behavior of a solution under a scenario can be measured by different criteria of robustness. Thus for the same conventional (non-robust) optimization problem one can construct several robust problem variants, which can be more or less difficult to solve. There are three popular basic criteria of robustness, and according to [19] they are called absolute robustness, robust deviation, and relative robust deviation. In some other publications, e.g. [1], the same criteria are referred to as max-min (min-max), min-max regret, and relative min-max regret, respectively.

Further, more modern and practical criteria are called OWA criteria. Their multi-objective function, which involves convex combination of conventional objective functions for each scenario, allows giving preferences to some scenarios. For particular combinations of coefficients, an OWA criterion becomes one of the three basic criteria. Before formal definitions of different robust solutions, we give some basic notation.

Let $S$ denote the set of all scenarios, and suppose that each scenario is encoded as an $n$-tuple, where $n$ is the number of input parameters. Let $X$ be a feasible solution, $F(X,s)$ the (conventional) objective-function value for solution $X$ under scenario $s$, $F_s^*$ the optimal (conventional) solution value for scenario $s$, and $\Phi$ the set of all solutions that are feasible for all scenarios. Suppose also that the considered conventional problem is a maximization problem, and that for any scenario $s$ the value $F_s^*$ is not zero. Then the three previously mentioned robustness criteria, i.e. their solutions are defined in the following way.

- *Absolute robust solution (max-min).* $X_A$ is a feasible solution whose minimum objective function value, measured over all scenarios, is as large as possible, i.e.

$$\mathrm{opt}_A = \max_{X\in\Phi}\min_{s\in S} F(X,s) = \max_{X\in\Phi} F_A(X) = F_A(X_A).$$

- *Robust deviation solution (min-max regret).* $X_D$ is a feasible solution whose maximum deviation from the conventional optimum, measured over all scenarios, is as small as possible, i.e.

$$\mathrm{opt}_D = \min_{X\in\Phi}\max_{s\in S}\Big(F_s^* - F(X,s)\Big) = \min_{X\in\Phi} F_D(X) = F_D(X_D).$$

- *Relative robust deviation solution (relative min-max regret).* $X_R$ is a feasible solution whose maximum relative deviation from the conventional optimum, measured over all scenarios, is as small as possible, i.e.

$$\mathrm{opt}_R = \min_{X\in\Phi}\max_{s\in S}\left(\frac{F_s^* - F(X,s)}{F_s^*}\right) = \min_{X\in\Phi} F_R(X) = F_R(X_R).$$

Here, $F_A$, $F_D$ and $F_R$ denote the objective functions that correspond to our three robustness criteria.

Apart from these three basic criteria of robustness, their more complex counterparts are also considered, which are called OWA criteria. Suppose that our collection of scenarios $S$ consists of $p$ scenarios denoted with $s_1, s_2, \ldots, s_p$. Any OWA criterion is based on a vector of real coefficients $a_1, a_2, \ldots, a_p$ given in advance, such that

$$0 \leq a_i \leq 1 \quad i = 1, 2, \ldots, p,$$
$$\sum_{i=1}^{p} a_i = 1$$

According to [17], the criterion is constructed as follows:
For a chosen solution $X$ the values $F(X,s)$ or the corresponding (relative) regrets are sorted in ascending or descending order, depending on the wanted generalization. OWA$_A$ variant uses ascending order, i.e. a permutation $\sigma$ is found such that

$$F(X, s_{\sigma(1)}) \leq F(X, s_{\sigma(2)}) \leq \ldots \leq F(X, s_{\sigma(p)}).$$

Then, the *OWA$_A$* cost for $X$ is computed as:

$$O_A(X) = \sum_{i=1}^{p} a_i \cdot F(X, s_{\sigma(i)}).$$

Similarly, OWA$_D$ and OWA$_R$ use descending order, i.e. a permutation $\sigma$ is found such that

$$F^*_{s_{\sigma(1)}} - F(X, s_{\sigma(1)}) \geq F^*_{s_{\sigma(2)}} - F(X, s_{\sigma(2)}) \geq \ldots \geq F^*_{s_{\sigma(p)}} - F(X, s_{\sigma(p)})$$
$$\frac{F^*_{s_{\sigma(1)}} - F(X, s_{\sigma(1)})}{F^*_{s_{\sigma(1)}}} \geq \frac{F^*_{s_{\sigma(2)}} - F(X, s_{\sigma(2)})}{F^*_{s_{\sigma(2)}}} \geq \cdots \geq \frac{F^*_{s_{\sigma(p)}} - F(X, s_{\sigma(p)})}{F^*_{s_{\sigma(p)}}},$$

respectively. Further, OWA$_D$ and OWA$_R$ *costs* for $X$ are computed:

$$O_D(X) = \sum_{i=1}^{p} a_i \cdot (F^*_{s_{\sigma(i)}} - F(X, s_{\sigma(i)}))$$
$$O_R(X) = \sum_{i=1}^{p} a_i \cdot \frac{F^*_{s_{\sigma(i)}} - F(X, s_{\sigma(i)})}{F^*_{s_{\sigma(i)}}}.$$

Then the three previously mentioned OWA criteria i.e. their solutions are defined in the following way:

- *OWA$_A$ solution*. $X_{OA}$ is a feasible solution which maximizes the function $O_A(X)$ over the whole collection of possible solutions, i.e.

$$\text{opt}_{OA} = O_A(X_{OA}) = \max_{X \in \Phi} O_A(X).$$

- *OWA$_D$ solution.* $X_{OD}$ is a feasible solution which minimizes the function $O_D(X)$ over the whole collection of possible solutions, i.e.

$$\text{opt}_{OD} = O_D(X_{OD}) = \min_{X \in \Phi} O_D(X).$$

- *OWA$_R$ solution.* $X_{OR}$ is a feasible solution which minimizes the function $O_R(X)$ over the whole collection of possible solutions, i.e

$$\text{opt}_{OR} = O_R(X_{OR}) = \min_{X \in \Phi} O_R(X).$$

With $a_1 = 1$, $a_2 = a_3 = \cdots = a_p = 0$, OWA$_A$ reduces to the traditional absolute robustness criterion. Similary, with $a_1 = 1$, $a_2 = a_3 = \cdots = a_p = 0$, OWA$_D$ and OWA$_R$ reduce to min-max regret and relative min-max regret, respectively.

## 1.2. Pareto efficiency

As an alternative to satisfying the above robustness criteria, there is another method for solving robust problems, which is based on so-called Pareto efficiency [9]. According to that method, solving a problem instance means finding not only one robustly optimal solution, but the whole collection of efficient solutions. To explain the concept of efficiency, we must first explain the related concept of domination.

- A solution $Z$ of a robust problem instance is *dominated* by another solution $\tilde{Z}$ if $Z$ is equally good or worse than $\widetilde{Z}$ under any scenario, and strictly worse than $\widetilde{Z}$ under at least one scenario.

- A solution $Z$ is *efficient* if it is not dominated by any other solution.

It is easy to show that for any of the three robustness criteria, either in its basic or in its OWA form, there exists an efficient solution that is robustly optimal according to that criterion [1,17]. Consequently, the method for solving a robust problem based on finding all efficient solutions can be regarded as more comprehensive than a method based on a particular criterion.

## 1.3. Interval uncertainty

An alternative to discrete set od scenarios is interval uncertainty. Rather then encoding each scenario with $n$-touple, where $n$ is the number of input parameters, possible parameter values are expressed by intervals. More precisely, we assume that the weight of a parameter $v_i$ can take any value from a given integer interval $I_i = [l_i, u_i]$. Paremeter values are chosen independently one from another. Thus the set of scenarios $S$ is implicitly given as the full Cartesian product of all intervals $I_i$. Such scenario set can be combined with any of the previously considered robustness criteria.

## 1.4. Maximum weighted independet set problem

As already announced, in this work we study the maximum weighted independent set problem. Its conventional variant is very well known and treated in many textbooks, e.g. [13, 15].

From the application point of view, interesting independent sets are those with large weights (interpreted as profits). This is a motivation for the following definitions.

- Let $G = (V, E)$ be an undirected graph, where $V$ is the set of vertices and $E$ the set of edges. An *independent set* of $G$ is a subset $X$ of $V$ such that no two vertices in $X$ are adjacent (connected by an edge from $E$).
- Let $G = (V, E)$ be an undirected graph whose vertices have weights. Suppose that all weights are nonnegative integers. A *maximum weighted independent set* of $G$ is an independent set of $G$ whose sum of vertex weights is as large as possible.
- The problem of finding a maximum weighted independent set in a given weighted graph is called the (conventional) *maximum weighted independent set problem* (the *MWIS problem*).

## 1.5. Minimum weighted vertex cover problem

A related problem to the MWIS problem is the minimum weighted vertex cover problem. Now, from the application point of view, interesting vertex covers should have small weights (interpreted as costs). This is a motivation for the following definitions.

- Let $G = (V,E)$ be an undirected graph, where $V$ is the set of vertices and $E$ the set of edges. A *vertex cover* of $G$ is a subset $Y$ of $V$ such that at least one endpoint of every edge from $E$ is in $Y$.

- Let $G = (V,E)$ be an undirected graph whose vertices have weights. Suppose that all weights are nonnegative integers. A *minimum weighted vertex cover* of $G$ is a vertex cover of $G$ whose sum of vertex weights is as small as possible.

- The problem of finding a minimum weighted vertex cover in a given weighted graph is called the (conventional) *minimumn weighted vertex cover problem* (the *MWVC problem*).

The relationship between independent sets and vertex covers in unweighted graphs is established by the following two claims, which are well known and easy to prove:

- *Let $X \subset V$ be an independent set. Then the complement $Y = V \setminus X$ is a vertex cover.*

- *Let $Y \subset V$ be a vertex cover. Then the complement $X = V \setminus Y$ is an independent set.*

Similar equivalences, which are also easy to prove, hold for weighted graphs:

- Let $X^*$ be an optimal solution for the MWIS problem (i.e. an independent set with maximum weight). Then its complement $Y^* = V \setminus X^*$ is an optimal solution for the MWVC problem (i.e. a vertex cover with minimum weight).

- Let $Y^*$ be an optimal solution for the MWVC problem (i.e. a vertex cover with minimum weight). Then its complement $X^* = V \setminus Y^*$ is an optimal solution for the MWIS problem (i.e. an independent set with maximum weight).

The weight of an independent set $X$, denoted as $F(X)$, is defined as the sum of weights of all vertices belonging to $X$. Similarly, the weight of a vertex cover $Y$, is denoted as $\bar{F}(Y)$ and it is the sum of weights of all vertices belonging to $Y$. Let $T$ be the total sum of weights of all vertices in $G$. If $X$ and $Y$ are complements, then it obviously holds that

$$F(X) + \bar{F}(Y) = T.$$

Denote with $F^*$ the optimal weight for the MWIS problem, and with $\bar{F}^*$ the optimal weight for the MWVC problem. Then the last two assertions guarantee that

$$F^* + \bar{F}^* = T.$$

As we can see, the conventional MWIS and MWVC problems are *equivalent*. Indeed, from the optimal solution of one problem, by computing a set complement, one can obtain the optimal solution of the other problem, and vice versa. Any algorithm [18] that solves one problem can be used for solving the other problem.

# 2. Relationships between robust variants of the MWIS and MWVC problem

The aim of this chapter is to explore relationships among robust variants of the two considered optimization problems. Or in other words, the aim is to find out whether the complement of a robustly optimal independent set must be a robustly optimal vertex cover, and vice-versa. We expect that the answer to this question might not be simple, e.g. it could depend on the chosen criterion of robustness.

## 2.1. The case of absolute robustness

Due to more scenarios, it is also necessary to redefine the previously defined MWIS and MWVC problem, i.e. instead of their conventional variants their robust variants should be considered. The weight of an independent set $X$ under the scenario $s$ will now be denoted as $F(X,s)$. Similarly, the weight of a vertex cover $Y$ under the scenario $s$ is denoted as $\bar{F}(Y,s)$. Let $T_s$ be the total sum of weights of all vertices of $G$ under the scenario $s$. Suppose that $X$ and $Y$ are complements. Then for each $s \in S$ it holds that

$$F(X,s) + \bar{F}(Y,s) = T_s.$$

Further, the symbol $F_s^*$ denotes the optimal solution weight for the (conventional) MWIS problem instance with vertex weights set according to the scenario $s \in S$. Similarly, $\bar{F}_s^*$ is the optimal solution weight for the (conventional) MWVC problem instance corresponding to the scenario $s$. Obviously, it holds:

$$F_s^* + \bar{F}_s^* = T_s.$$

However, such redefinition can be done in several ways. In this section we restrict to the variants obtained by applying the absolute criterion of robustness. According to the general rule of absolute robustness from the previous section, the following definitions are obtained.

**Definition 2.1.** An absolute robust solution for the MWIS problem is an independent set $X_A$ that maximizes the function $\min_{s\in S} F(X,s)$ over the whole collection of possible independent sets $X$.

**Definition 2.2.** An absolute robust solution for the MWVC problem is a vertex cover $Y_A$ that minimizes the function $\max_{s\in S} \bar{F}(Y,s)$ over the whole collection of possible vertex covers $Y$.

A natural question one would like to answer is whether the absolute robust MWIS problem is equivalent to the absolute robust MWVC problem, as it was true in the conventional case. More precisely: is the complement of a robustly optimal independent set a robustly optimal vertex cover, and vice-versa? A partial answer to this question is given by the following proposition.

**Proposition 2.3.** Suppose that all scenarios have the same sum of vertex weights, i.e. $T_s = T$ for all $s \in S$. Then the complement of an absolute robust solution for the MWIS problem is an absolute robust solution for the MWVC problem, and vice-versa.

*Proof.* Let $X_A$ be an absolute robust solution for the MWIS problem, and let $Y_A$ be the complement of $X_A$. Then we have:

$$\begin{aligned}
\min_{s\in S} F(X,s) &\quad \text{achieves maximum for} \quad X = X_A \quad \Longrightarrow \\
\min_{s\in S} (T - \bar{F}(Y,s)) &\quad \text{achieves maximum for} \quad Y = Y_A \quad \Longrightarrow \\
T - \max_{s\in S} \bar{F}(Y,s) &\quad \text{achieves maximum for} \quad Y = Y_A \quad \Longrightarrow \\
\max_{s\in S} \bar{F}(Y,s) &\quad \text{achieves minimum for} \quad Y = Y_A.
\end{aligned}$$

Thus $Y_A$ is by definition an absolute robust solution for the MWVC problem. The proof in opposite direction is conducted analogously. ■

Note that Proposition 2.3. assures equivalence among the considered robust problems only in a special case, i.e. when all scenarios have the same total sum of vertex weights. Unfortunately, such equivalence does not hold in general, as demonstrated by the following example.

**Example 2.4.** Let us consider the graph with nine vertices shown in Figure 2.1. There are three scenarios for weights, as indicated by triple labels assigned to vertices. Then the corresponding absolute robust solutions for the MWIS and MWVC problem, respectively, are presented in Table 2.1. The left-hand side of the table comprises all nontrivial independent sets, i.e. those that cannot be extended by adding more vertices. Similarly, the right-hand side of the table contains all nontrivial vertex covers, i.e. those that cannot be reduced by removing some of their vertices. For each independent set or vertex cover, there is a list of its weights under different scenarios. In each list, the worst weight is underlined. Robustly optimal solutions (i.e. those whose worst weight is the best) are shown in boldface. We can check that the complement of the robustly optimal independent set is not a robustly optimal vertex cover. Similarly, the complement of the robustly optimal vertex cover is not a robustly optimal independent set. The found absolute robust solutions are also shown in Figure 2.1 by shading. Black vertices belong to the optimal independent set, light grey vertices are from the optimal vertex cover, while dark gray vertices are common to both sets.

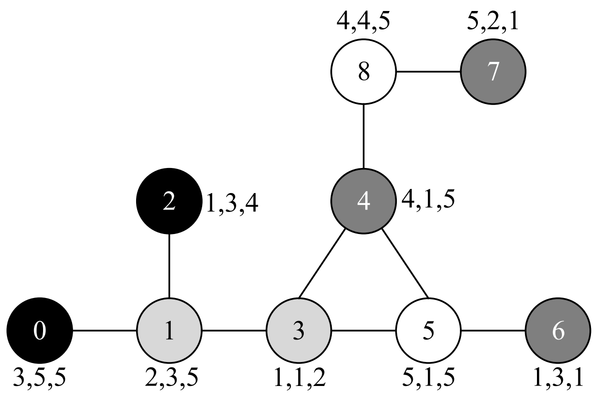

Figure 2.1: A graph where the complement of an absolute robust solution for the MWIS problem is not an absolute robust solution for the MWVC problem

In the remaining part of this section, we will explain how our Proposition 2.3 can be used to transfer some of the available complexity or approximation results from the robust MWIS to the robust MWVC context. This can be done although Proposition 1 establishes only a partial equivalence among absolutely robust MWIS and MWVC problems.

Indeed, in [30] there are two results on NP-hardness of the absolute robust MWIS problem on a special class of graphs called interval graphs. Both results can be converted to the corresponding MWVC problem. Conversion is done so that the MWIS problem instances used in the proofs are (polynomially) reduced to the corresponding MWVC instances. Such reduction is possible thanks to the fact that in both proofs the constructed instances satisfy the restriction regarding scenarios from Proposition 2.3.

Table 2.1: Finding absolute robust solutions for the graph shown in Figure 2.1

| Independent set | Weight for each scenario | | | Vertex cover | Weight for each scenario | | |
|---|---|---|---|---|---|---|---|
| 0,2,3,6,7 | $\underline{11}$ | 14 | 13 | 1,4,5,8 | 15 | 9 | $\underline{20}$ |
| 0,2,3,6,8 | $\underline{10}$ | 16 | 17 | 1,4,5,7 | $\underline{16}$ | 7 | 16 |
| **0,2,4,6,7** | $\underline{14}$ | 14 | 16 | 1,3,5,8 | 12 | 9 | $\underline{17}$ |
| 0,2,5,7 | 14 | $\underline{11}$ | 15 | 1,3,4,6,8 | 12 | 12 | $\underline{18}$ |
| 0,2,5,8 | $\underline{13}$ | 13 | 19 | **1,3,4,6,7** | 13 | 10 | $\underline{14}$ |
| 1,4,6,7 | 12 | $\underline{9}$ | 12 | 0,2,3,5,8 | 14 | 14 | $\underline{21}$ |
| 1,5,7 | 12 | $\underline{6}$ | 11 | 0,2,3,4,6,8 | 14 | 17 | $\underline{22}$ |
| 1,5,8 | 11 | $\underline{8}$ | 15 | 0,2,3,4,6,7 | 15 | 15 | $\underline{18}$ |

In [30] there is also a pseudo-polynomial-time algorithm for solving the absolute robust MWIS problem on interval graphs. Obviously, the same algorithm can also be used to solve the corresponding MWVC problem. More precisely, for a given problem instance, a robustly optimal independent set is first found and then converted (in polynomial time) into the complementary vertex cover. Such computation will be correct if the given instance satisfies the restriction from Proposition 2.3.

In addition to the results from [30] there is a fairly general proposition in [1] dealing with approximability of robust solutions within the number of scenarios. The proposition is applicable to absolute robust variants originating from conventional minimization, so that it can be applied to the MWVC problem, but not to the MWIS problem.

Putting it all together, our Proposition 2.3 combined with the results from [1, 30] brings the following consequences. They deal with interval graphs, where the conventional MWVC problem (being equivalent to the conventional MWIS problem) is solvable in polynomial time.

**Corollary 2.5.** We consider the absolute robust MWVC problem on interval graphs. Then the considered problem is NP-hard even with only two scenarios. An instance of the problem whose scenarios have the same sum of vertex weights can be solved in pseudo-polynomial time when the number of scenarios is bounded. The problem is strongly NP-hard when the number of scenarios is unbounded, but approximable within the number of scenarios.

## 2.2. The case of robust deviation

Similarly as in the previous section, we again study robust variants of the MWIS and MWVC problem with explicitly given scenarios. But now we apply the second criterion of robustness called robust deviation. The general rule of robust deviation has been stated in Chapter 1. By applying that rule to our problems, the following two definitions are obtained.

**Definition 2.6.** A robust deviation solution for the MWIS problem is an independent set $X_D$ that minimizes the function $\max_{s\in S}(F_s^* - F(X,s))$ over the whole collection of possible independent sets $X$.

**Definition 2.7.** A robust deviation solution for the MWVC problem is a vertex cover $Y_D$ that minimizes the function $\max_{s\in S}(\bar{F}(Y,s) - \bar{F}_s^*)$ over the whole collection of possible vertex covers $Y$.

$F_s^*$ and $\bar{F}_s^*$ are the optimal weights for the MWIS problem and the MWVC problem under scenario $s$, respectively.

Again as in the previous section, an important question is whether the obtained robust MWIS and MWVC problem are equivalent or not. This time the answer is affirmative, and it is given by the following Proposition 2.8.

**Proposition 2.8.** The complement of a robust deviation solution for the MWIS problem is a robust deviation solution for the MWVC problem, and vice-versa.

*Proof.* Let $X_D$ be a robust deviation solution for the MWIS problem, and let $Y_D$ be the complement of $X_D$. Then:

$$\begin{aligned}
\max_{s\in S}(F_s^* - F(X,s)) &\quad \text{achieves minimum for} \quad X = X_D \implies \\
\max_{s\in S}(T_s - \bar{F}_s^* - (T_s - \bar{F}(Y,s)) &\quad \text{achieves minimum for} \quad Y = Y_D \implies \\
\max_{s\in S}(\bar{F}(Y,s) - \bar{F}_s^*) &\quad \text{achieves minimum for} \quad Y = Y_D.
\end{aligned}$$

Thus $Y_D$ is by definition a robust deviation solution for the MWVC problem. The claim in the +opposite direction is proved analogously. ■

**Example 2.9.** Let us consider again the graph from Figure 2.1. By scanning and recomputing the data from Table 2.1, one can easily obtain Table 2.2, where the corresponding robust deviation solutions are presented. For each independent set or vertex cover, Table 2.2 shows the list of its deviations from the conventional optimum under different scenarios. In each list the largest deviation is underlined. Robustly optimal solutions (i.e. those whose largest deviation is minimal) are shown in boldface. So we see that there are two robustly optimal independent sets, consisting of vertices 0,2,4,6,7 and 0,2,5,8, respectively. Also, there are two robustly optimal vertex covers, comprising vertices 1,3,5,8 and 1,3,4,6,7, respectively. The complements of the found independent sets coincide with the found vertex covers, and vice-versa. The robust objective-function value is always 3.

Table 2.2: Finding robust deviation solutions for the graph shown in Figure 2.1

| Independent set | Deviation for each scenario | | | Vertex cover | Deviation for each scenario | | |
|---|---|---|---|---|---|---|---|
| 0,2,3,6,7 | 3 | 2 | $\underline{6}$ | 1,4,5,8 | 3 | 2 | $\underline{6}$ |
| 0,2,3,6,8 | $\underline{4}$ | 0 | 2 | 1,4,5,7 | $\underline{4}$ | 0 | 2 |
| **0,2,4,6,7** | 0 | 2 | $\underline{3}$ | **1,3,5,8** | 0 | 2 | $\underline{3}$ |
| 0,2,5,7 | 0 | $\underline{5}$ | 4 | 1,3,4,6,8 | 0 | $\underline{5}$ | 4 |
| **0,2,5,8** | 1 | $\underline{3}$ | 0 | **1,3,4,6,7** | 1 | $\underline{3}$ | 0 |
| 1,4,6,7 | 2 | $\underline{7}$ | 7 | 0,2,3,5,8 | 2 | $\underline{7}$ | 7 |
| 1,5,7 | 2 | $\underline{10}$ | 8 | 0,2,3,4,6,8 | 2 | $\underline{10}$ | 8 |
| 1,5,8 | 3 | $\underline{8}$ | 4 | 0,2,3,4,6,7 | 3 | $\underline{8}$ | 4 |

By using Proposition 2.8, many complexity or approximation results on the robust deviation MWIS problem can be reinterpreted for the MWVC problem. Indeed, [30] contains two NP-hardness results and one pseudo-polynomial-time algorithm for the robust deviation MWIS problem on interval graphs. In [16] there are additional approximability results dealing with the same problem again on interval graphs. All those results can be converted to the corresponding MWVC problem. Conversion is done in the same manner as explained in Section 2.1. Switching from MWIS to MWVC does not spoil accuracy of approximation thanks to the following fact (visible in the proof of Proposition 2.8): two complementary vertex sets (i.e. an independent set an the corresponding vertex cover) have the same "regret" over any scenario, and therefore their robust objective function values are also the same.

Putting it all together, by combining [16, 30] with Proposition 2.8, the following results can be established. They again deal with interval graphs where the conventional MWVC problem is solvable in polynomial time.

**Corollary 2.10.** We consider the robust deviation MWVC problem on interval graphs. Then the considered problem is NP-hard even with only two scenarios, and it can be solved in pseudo-polynomial time when the number of scenarios is bounded. The problem also admits a fully polynomial approximation scheme if the number of scenarios is bounded. The problem is strongly NP-hard when the number of scenarios is unbounded, but approximable within the number of scenarios.

## 2.3. The case of relative robust deviation

In this section we explore robust variants of the MWIS and MWVC problem based on explicitly given scenarios and on the third criterion of robustness called relative robust deviation. By applying the general formulation of the criterion from Chapter 1 to our problems, the next two definitions are obtained.

**Definition 2.11.** A relative robust deviation solution for the MWIS problem is an independent set $X_R$ that minimizes the function $\max_{s \in S}((F_s^* - F(X,s))/F_s^*)$ over the whole collection of possible independent sets $X$.

**Definition 2.12.** A relative robust deviation solution for the MWVC problem is a vertex cover $Y_R$ that minimizes the function $\max_{s \in S}((\bar{F}(Y,s) - \bar{F}_s^*)/\bar{F}_s^*)$ over the whole collection of possible vertex covers $Y$.

It is assumed here that both $F_s^*$ and $\bar{F}_s^*$ are $> 0$. This is in fact always the case except for trivial problem instances.

Again as in the previous sections, it would be interesting to determine whether the newly obtained robust MWIS and MWVC problem are equivalent or not. The following proposition establishes a sufficient condition for equivalence.

**Proposition 2.13.** Suppose that the ratio among optimal solution weights for the conventional MWIS and MWVC problem, respectively, is the same for all scenarios. Or in other words, suppose that $\bar{F}_s^*/F_s^* = Q$ for all $s \in S$. Then the complement of a relative robust deviation solution for the MWIS problem is a relative robust deviation solution for the MWVC problem, and vice-versa.

*Proof.* Let $X_R$ be a relative robust deviation solution for the MWIS problem, and let $Y_R$ be the complement of $X_R$. Then:

$$\begin{aligned}
\max_{s\in S}\left(\frac{F_s^*-F(X,s)}{F_s^*}\right) &\quad \text{achieves minimum for} \quad X=X_R \quad \Longrightarrow \\
\max_{s\in S}\left(\frac{T_s-\bar{F}_s^*-T_s+\bar{F}(Y,s)}{F_s^*}\right) &\quad \text{achieves minimum for} \quad Y=Y_R \quad \Longrightarrow \\
\max_{s\in S}\left(\frac{\bar{F}_s^*}{F_s^*}\cdot\frac{\bar{F}(Y,s)-\bar{F}_s^*}{\bar{F}_s^*}\right) &\quad \text{achieves minimum for} \quad Y=Y_R \quad \Longrightarrow \\
Q\cdot\max_{s\in S}\left(\frac{\bar{F}(Y,s)-\bar{F}_s^*}{\bar{F}_s^*}\right) &\quad \text{achieves minimum for} \quad Y=Y_R \quad \Longrightarrow \\
\max_{s\in S}\left(\frac{\bar{F}(Y,s)-\bar{F}_s^*}{\bar{F}_s^*}\right) &\quad \text{achieves minimum for} \quad Y=Y_R.
\end{aligned}$$

Thus $Y_R$ is by definition a relative robust deviation solution for the MWVC problem. The proof in the opposite direction is analogous. ■

Note that Proposition 2.13 guarantees equivalence among the considered robust problems only under some very special conditions. Unfortunately, equivalence is not assured in general. It is also not assured by the condition from Proposition 2.3 (equal total sum of weights for all scenarios). Indeed, here follows an example.

**Example 2.14.** We consider the graph shown in Figure 2.2. The corresponding absolute robust solutions are presented in Table 2.3, which is organized analogously as Table 2.1. We can see that the two solutions are complementary one to another - this is in accordance with Proposition 2.3, which can be applied since all scenarios have the same total sum of weights. By scanning and recomputing the data from Table 2.3, we obtain Table 2.4, where the corresponding relative robust deviation solutions are determined. For each independent set or vertex cover, Table 2.4 shows its relative deviations from the conventional optimum depending on scenarios. The largest relative deviations are underlined. Robustly optimal solutions (i.e. those whose largest relative deviation is minimal) are shown in boldface. We can check that the complement of the robustly optimal independent set is not a robustly optimal vertex cover. Similarly, the complement of the robustly optimal vertex cover is not a robustly optimal independent set. The found relative robust deviation solutions are shown in Figure 2.2 by shading. The same colors are used as in Example 2.4.

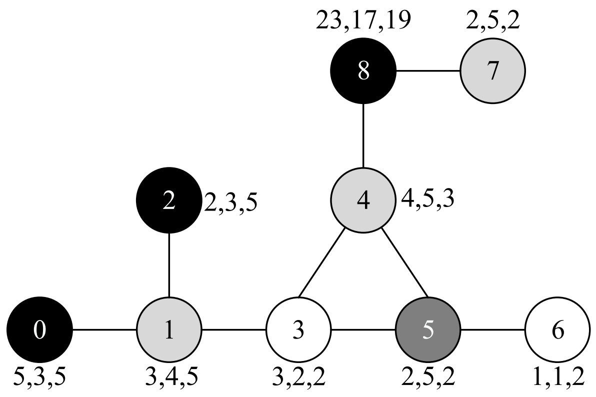

Figure 2.2: A graph where the complement of a relative robust deviation solution for the MWIS problem is not a relative robust deviation solution for the MWVC problem

Table 2.3: Finding absolute robust solutions for the graph shown in Figure 2.2

| Independent set | Weight for each scenario | | | Vertex cover | Weight for each scenario | | |
|---|---|---|---|---|---|---|---|
| 0,2,3,6,7 | 13 | 14 | 16 | 1,4,5,8 | 32 | 31 | 29 |
| 0,2,3,6,8 | 34 | 26 | 33 | 1,4,5,7 | 11 | 19 | 12 |
| 0,2,4,6,7 | 14 | 17 | 17 | 1,3,5,8 | 31 | 28 | 28 |
| 0,2,5,7 | 11 | 16 | 14 | 1,3,4,6,8 | 34 | 29 | 31 |
| **0,2,5,8** | 32 | 28 | 31 | **1,3,4,6,7** | 13 | 17 | 14 |
| 1,4,6,7 | 10 | 15 | 12 | 0,2,3,5,8 | 35 | 30 | 33 |
| 1,5,7 | 7 | 14 | 9 | 0,2,3,4,6,8 | 38 | 31 | 36 |
| 1,5,8 | 28 | 26 | 26 | 0,2,3,4,6,7 | 17 | 19 | 19 |

Table 2.4: Finding relative robust deviation solutions for the graph shown in Figure 2.2

| Independent set | Relative deviation for each scenario | | | Vertex cover | Relative deviation for each scenario | | |
|---|---|---|---|---|---|---|---|
| 0,2,3,6,7 | 0.618 | 0.500 | 0.515 | 1,4,5,8 | 1.909 | 0.824 | 1.417 |
| 0,2,3,6,8 | 0.000 | 0.071 | 0.000 | **1,4,5,7** | 0.000 | 0.118 | 0.000 |
| 0,2,4,6,7 | 0.588 | 0.393 | 0.485 | 1,3,5,8 | 1.818 | 0.647 | 1.333 |
| 0,2,5,7 | 0.676 | 0.429 | 0.576 | 1,3,4,6,8 | 2.091 | 0.706 | 1.583 |
| **0,2,5,8** | 0.059 | 0.000 | 0.061 | 1,3,4,6,7 | 0.182 | 0.000 | 0.167 |
| 1,4,6,7 | 0.706 | 0.464 | 0.636 | 0,2,3,5,8 | 2.182 | 0.765 | 1.750 |
| 1,5,7 | 0.794 | 0.500 | 0.727 | 0,2,3,4,6,8 | 2.455 | 0.824 | 2.000 |
| 1,5,8 | 0.176 | 0.071 | 0.212 | 0,2,3,4,6,7 | 0.545 | 0.118 | 0.583 |

## 2.4. Interval uncertainty

This section studies situations where uncertainty in vertex weights is expressed by intervals. More precisely, we assume that the weight of a vertex $v_i$ can take any value from a given integer interval $I_i = [l_i, u_i]$. As previously stated, such a scenario set can be combined with any of the previously considered robustness criteria. In this way, robust variants of the MWIS and MWVC problem are obtained, which are special cases of those from Sections 2.1, 2.2 and 2.3.

Again, we could ask ourselves whether the obtained robust problem variants are equivalent in the sense that the complement of a robustly optimal independent set is a robustly optimal vertex cover, and vice-versa. Obviously, the answer depends on the chosen robustness criteria, and it should probably be the same or similar as in Section 2.1, 2.2 or 2.3. However, since the scenario set is rather regular, it is possible that the answer could be somewhat different of more specific. In this section we will explore such possibilities.

Let us first consider the MWIS and MWVC problem variants based on interval uncertainty and *absolute robustness*. Let us identify two special scenarios. The *minimum scenario* is the one where each vertex $v_i$ has the minimum possible weight $l_i$. Similarly, the *maximum scenario* is the one where each vertex $v_i$ has the maximum possible weight $u_i$. It is easy to see that the following claim is valid.

**Proposition 2.15.** Suppose that uncertainty in vertex weights is given by intervals. Then an absolute robust solution for the MWIS problem is obtained by solving the conventional MWIS problem according to the minimum scenario. Similarly, an absolute robust solution for the MWVC problem is obtained by solving the conventional MWVC problem according to the maximum scenario.

*Proof.* The claim is an obvious consequence of the way how absolute robustness is defined, combined with the fact that the minimum and maximum scenarios are available. The same claim in a more general setting is also mentioned in [1]. More formal proof for the MWIS problem is given in [30]. The same proof can easily be adjusted to the MWVC problem. ■

According to Proposition 2.15, the considered absolute robust problem variants with interval uncertainty can be solved relatively easily, i.e. as conventional variants. But note that those conventional variants are in general still NP-hard. However, there are some graph types that allow polynomial-time algorithms [7, 16, 30]. Thus the following consequence of Proposition 2.15

can be stated.

**Corollary 2.16.** Let the involved graph be an apple-free graph, an interval graph or a tree. Then an absolute robust solution for the MWIS or MWVC problem with interval uncertainty can be obtained in polynomial time.

Note also that Proposition 2.15 does not claim that solutions of the considered problems are equivalent in the sense that one of them is the complement of the other. Indeed, according to Proposition 2.15, each problem should be solved separately by using a different scenario. This point is illustrated by the following example.

**Example 2.17.** Let us consider the graph in Figure 2.3 whose vertex weights are given by intervals. Then the corresponding absolute robust solutions for the MWIS and MWVC problem, respectively, are shown in Table 2.5. For each independent set Table 2.5 shows its weight according to the minimum scenario. The robustly optimal independent set (i.e. the one with the largest weight) is shown in boldface. Similarly, for each vertex cover Table 2.5 shows its weight according to the maximum scenario. The robustly optimal vertex cover (i.e. the one with the smallest weight) is shown in boldface. We see that the complement of the optimal independent set is not an optimal vertex cover. Similarly, the complement of the optimal vertex cover is not an optimal independent set. The found solutions are shown in Figure 2.3 by shading.

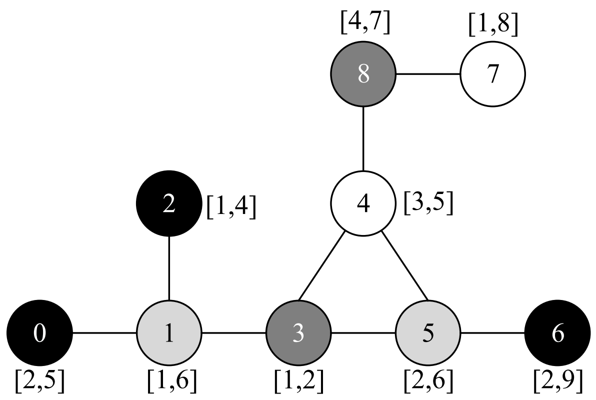

Figure 2.3: A graph where the complement of an optimal solution for the MWIS problem under the minimum scenario is not an optimal solution for the MWVC problem under the maximum scenario

Table 2.5: Finding absolute robust solutions for the graph shown in Figure 2.3

| Independent set | Weight under minimum scenario | Vertex cover | Weight under maximum scenario |
|---|---|---|---|
| 0,2,3,6,7 | 7 | 1,4,5,8 | 24 |
| **0,2,3,6,8** | 10 | 1,4,5,7 | 25 |
| 0,2,4,6,7 | 9 | **1,3,5,8** | 21 |
| 0,2,5,7 | 6 | 1,3,4,6,8 | 29 |
| 0,2,5,8 | 9 | 1,3,4,6,7 | 30 |
| 1,4,6,7 | 7 | 0,2,3,5,8 | 24 |
| 1,5,7 | 4 | 0,2,3,4,6,8 | 32 |
| 1,5,8 | 7 | 0,2,3,4,6,7 | 33 |

One way to enforce equivalence among the above considered MWIS and MWVC problem variants would be imposing an additional constraint. Such constraint would require that the sum of vertex weights in each scenario is equal to a predefined value $T$. The obtained (restricted) uncertainty set can be visualized as an intersection of a hyper-parallelepiped (Cartesian product) and a hyperplane. The restriction makes sense in applications where scenarios describe different possibilities to distribute a fixed amount of some resource. With the restricted uncertainty set, the equivalence of the two problems is assured by Proposition 2.3. On the other hand, the solutions from Proposition 2.15 become infeasible since the hyperplane cuts off both the minimum and the maximum scenario.

Let us now consider the MWIS and MWVC problem variants based on interval uncertainty and *robust deviation*. Again we have to identify a special type of scenario. An *extremal scenario* is such a scenario where each vertex $v_i$ has either minimal or maximal value (either $l_i$ or $u_i$). The importance of extremal scenarios is presented by the following proposition.

**Proposition 2.18.** Suppose that uncertainty in vertex weights is expressed by intervals. Then a robust deviation solution for the MWIS problem can be obtained by using a reduced scenario set consisting only of extremal scenarios. The same claim is also valid for the MWVC problem. Moreover, the complement of a robust deviation solution for the MWIS problem is a robust deviation solution for the MWVC problem, and vice-versa.

*Proof.* The first claim dealing with the MWIS problem has been proved in [30]. The second claim dealing with the MWVC problem then follows from Proposition 2.8 which is applicable in the considered situation. The third claim only repeats the statement of Proposition 2.8. ■

Proposition 2.18 assures that the considered problem variants based on interval uncertainty and robust deviation can be solved a little bit more efficiently than expected, i.e. with a reduced set of scenarios. But note that the reduced set is still fairly large. Indeed, for a graph with $n$ vertices, there can be as many as $2^n$ extremal scenarios.

Proposition 2.18 can also serve for transforming some of the available results on the robust MWIS problem into similar results for the MWVC problem. Indeed, [16] contains an NP-hardness and an approximability theorem that both refer to the robust deviation MWIS problem with interval uncertainty on interval graphs. Both results can be converted to the corresponding MWVC problem. Such conversion is correct for the same reasons as already explained in Section 2.2. More precisely, we obtain the following corollary.

**Corollary 2.19.** We consider the robust deviation MWVC problem on interval graphs. Uncertainty in vertex weights in expressed by interval representation. Then the considered problem is NP-hard. Also, the problem is approximable within 2.

At the end of this section, let us say a few words about the MWIS and MWVC problem variants based on interval uncertainty and *relative robust deviation*. According to Section 2.3, one would expect that such variants are not equivalent, i.e. that their solutions are not complementary to one another. This is indeed true. In order to check it we have constructed an additional Example 2.20, which is similar to Example 2.14 but based on interval uncertainty.

**Example 2.20.** In our example, the graph consists of 9 vertices, and each interval consisted of 2 integers, so that the total number of implicitly given scenarios was $2^9 = 512$. To find robust solutions with so many scenarios, we employed the CPLEX software package [14]. In order to be solvable with CPLEX, a problem must be written in form of linear programming. How to translate different robust variants of the MWIS problem in linear programming form will be shown in Section 3.5. The obtained results confirmed our expectations, e.g. it turned out that the complement of the computed robustly optimal independent set is not a robustly optimal vertex cover. The found solutions are shown in Figure 2.4 by shading.

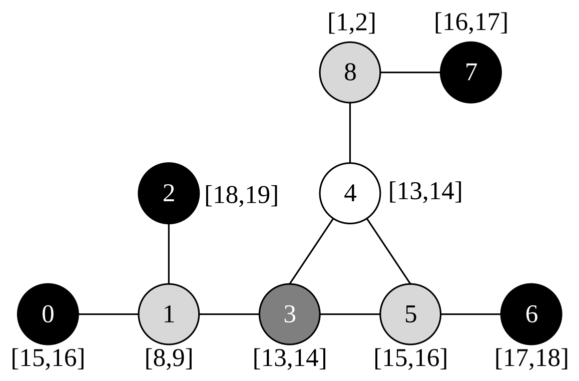

Figure 2.4: A graph where uncertainty in vertex weights is expressed by intervals. At he same time, the complement of a relative robust deviation solution for the MWIS problem is not a relative robust deviation solution for the MWVC problem

## 2.5. The case of OWA criteria

OWA criteria are generalizations of the three previously mentioned criteria. We expect for OWA criteria to have similar properties such as in Proposition 2.3, 2.8 and 2.13. First, we will put emphasis on the OWA criterion that extends absolute robustness.

Let us denote again the whole list of scenarios as $s_1, s_2, \dots, s_p$. For a chosen independent set $X$ the weights $F(X,s)$ are sorted in ascending order, i.e. a permutation $\sigma$ is found such that

$$F(X, s_{\sigma(1)}) \leq F(X, s_{\sigma(2)}) \leq \dots \leq F(X, s_{\sigma(p)}).$$

As stated in Chapter 1, with given coefficients $a_1, a_2, ..., a_p$, *OWA$_A$ cost* for $X$ is computed as:

$$O_A(X) = \sum_{i=1}^{p} a_i \cdot F(X, s_{\sigma(i)}).$$

This cost should be maximized for independent sets.

Similary, for a chosen vertex cover $Y$ the weights $\bar{F}(Y,s)$ are sorted in descending order, i.e. a permutation $\psi$ is found such that

$$\bar{F}(Y, s_{\psi(1)}) \geq \bar{F}(Y, s_{\psi(2)}) \geq \dots \geq \bar{F}(Y, s_{\psi(p)}).$$

With the same given coefficients $a_1, a_2, ..., a_p$, the *OWA$_A$ cost* for $Y$ is computed as:

$$\bar{O}_A(Y) = \sum_{i=1}^{p} a_i \cdot \bar{F}(Y, s_{\psi(i)}).$$

This cost should be minimized for vertex covers.

**Definition 2.21.** An OWA$_A$ solution for the MWIS problem is an independent set $X_{OA}$ that maximizes the function $O_A(X)$ over the whole collection of possible independent sets $X$.

**Definition 2.22.** An OWA$_A$ solution for the MWVC problem is a vertex cover $Y_{OA}$ that minimizes the function $\bar{O}_A(Y)$ over the whole collection of possible vertex covers $Y$.

Similarly as in the previous sections, we are concerned with the following question: is the OWA$_A$ variant of the MWIS problem equivalent to the OWA$_A$ variant of the MWVC problem? We assume that in both problems the coefficients $a_1, a_2, ..., a_p$ are chosen in the same way. Of course, we already know that the equivalence cannot hold in general - this has been shown by Example 2.4, which can be interpreted as an OWA$_A$ example where $a_1 = 1$ and $a_2 = a_3 = \dots = a_p = 0$. Still, there are some special cases where equivalence holds, as described by the following two propositions.

**Proposition 2.23.** Suppose that all scenarios have the same sum of vertex weights, i.e. $T_s = T$ for all $s \in S$. Then the complement of an OWA$_A$ solution for the MWIS problem is an OWA$_A$ solution for the MWVC problem, and vice-versa.

*Proof.* Let $X$ be any independent set and $Y$ its complement. Then it holds that $F(X,s) + \bar{F}(Y,s) = T$ for any scenario $s \in S$. Also, from

$$F(X, s_{\sigma(1)}) \leq F(X, s_{\sigma(2)}) \leq \ldots \leq F(X, s_{\sigma(p)}),$$

or equivalently

$$T - \bar{F}(Y, s_{\sigma(1)}) \leq T - \bar{F}(Y, s_{\sigma(2)}) \leq \ldots \leq T - \bar{F}(Y, s_{\sigma(p)}),$$

if follows that

$$\bar{F}(Y, s_{\sigma(1)}) \geq \bar{F}(Y, s_{\sigma(2)}) \geq \ldots \geq \bar{F}(Y, s_{\sigma(p)}).$$

Or in other words, the permutation $\sigma$ used by the OWA$_A$ criterion for an independent set coincides with the permutation $\psi$ needed for the complementary vertex cover. Consequently, if $X$ and $Y$ are complements, then it holds:

$$\begin{aligned} O_A(X) &= \sum_{i=1}^{p} a_i \cdot F(X, s_{\sigma(i)}) = \sum_{i=1}^{p} a_i (T - \bar{F}(Y, s_{\sigma(i)}) \\ &= T - \sum_{i=1}^{p} a_i \cdot \bar{F}(Y, s_{\psi(i)}) = T - \bar{O}_A(Y). \end{aligned}$$

Assume now that $X_{OA}$ is an OWA$_A$ solution for the MWIS problem, and that $Y_{OA}$ is the complement of $X_{OA}$. Then:

$$\begin{array}{rll} O_A(X) & \text{achieves maximum for} & X = X_{OA} \implies \\ T - \bar{O}_A(Y) & \text{achieves maximum for} & Y = Y_{OA} \implies \\ \bar{O}_A(Y) & \text{achieves minimum for} & Y = Y_{OA}. \end{array}$$

Thus $Y_{OA}$ is by definition an OWA$_A$ solution for the MWVC problem. The proof in opposite direction is conducted analogously. ■

**Proposition 2.24.** Suppose that all coefficients $a_1, a_2, \ldots, a_p$ are equal, i.e. $a_1 = a_2 = \cdots = a_p = 1/p$. Then the complement of an OWA$_A$ solution for the MWIS problem is an OWA$_A$ solution for the MWVC problem, and vice-versa.

*Proof.* Let $X$ be any independent set and $Y$ its complement. Then $F(X,s) + \bar{F}(Y,s) = T_s$ for any scenario $s \in S$. Denote the expression $(1/p)\sum_{i=1}^{p} T_i$ with $\tilde{T}$. Next it holds:

$$\begin{aligned} O_A(X) &= \frac{1}{p}\sum_{i=1}^{p} F(X, s_{\sigma(i)}) = \frac{1}{p}\sum_{i=1}^{p}(T_{s_{\sigma(i)}} - \bar{F}(Y, s_{\sigma(i)})) \\ &= \frac{1}{p}\sum_{i=1}^{p} T_{s_{\sigma(i)}} - \frac{1}{p}\sum_{i=1}^{p} \bar{F}(Y, s_{\sigma(i)}) \\ &= \tilde{T} - \frac{1}{p}\sum_{i=1}^{p} \bar{F}(Y, s_{\psi(i)}) \\ &= \tilde{T} - \bar{O}_A(Y). \end{aligned}$$

The rest of the proof is similar as for Proposition 2.23. ■

Note that Proposition 2.23 is a generalization of Proposition 2.3 from Section 2.1. Note also that Proposition 2.24 is not a surprise since it refers to the case where the OWA$_A$ variants of both problems reduce to their conventional variants over an *average* scenario. It is a scenario where the weight of any vertex is equal to the average of its weights over all scenarios. Reduction is correct because $O_A(X)$ (or $\bar{O}_A(Y)$) is a sum of sums, and the order of summation can be reversed. The involved conventional variants are surely equivalent even with non-integer vertex weights.

So far we have considered the OWA criterion that extends absolute robustness. Further, the OWA$_D$ criterion is a generalization of the traditional robust deviation criterion, i.e. OWA$_D$ with $a_1 = 1$, $a_2 = a_3 = \cdots = a_p = 0$ reduces to the traditional min-max regret. It can easily be seen that the OWA$_D$ variant of the MWIS problem is equivalent to the OWA$_D$ variant of the MWVC problem. More precisely, Proposition 2.8 from Section 2.2 can be generalized to cover OWA$_D$ instead of simple robust deviation.

For a chosen independent set $X$ and vertex cover $Y$ the respective regrets $F_s^* - F(X,s)$ and $\bar{F}(Y,s) - \bar{F}_s^*$ are sorted in descending order, i.e. permutations $\sigma$ and $\psi$ are found such that

$$F^*_{s_{\sigma(1)}} - F(X, s_{\sigma(1)}) \geq F^*_{s_{\sigma(2)}} - F(X, s_{\sigma(2)}) \geq \ldots \geq F^*_{s_{\sigma(p)}} - F(X, s_{\sigma(p)})$$
$$\bar{F}(Y, s_{\psi(1)}) - \bar{F}^*_{s_{\psi(1)}} \geq \bar{F}(Y, s_{\psi(2)}) - \bar{F}^*_{s_{\psi(2)}} \geq \ldots \geq \bar{F}(Y, s_{\psi(p)}) - \bar{F}^*_{s_{\psi(p)}}.$$

Then *OWA$_D$ cost* for $X$ and $Y$ are computed as:

$$O_D(X) = \sum_{i=1}^{p} a_i \cdot (F^*_{s_{\sigma(i)}} - F(X, s_{\sigma(i)}))$$
$$\bar{O}_D(Y) = \sum_{i=1}^{p} a_i \cdot (\bar{F}(Y, s_{\psi(i)}) - \bar{F}^*_{s_{\psi(i)}}).$$

**Definition 2.25.** An OWA$_D$ solution for the MWIS problem is an independent set $X_{OD}$ that minimizes the function $O_D(X)$ over the whole collection of possible independent sets $X$.

**Definition 2.26.** An OWA$_D$ solution for the MWVC problem is a vertex cover $Y_{OD}$ that minimizes the function $\bar{O}_D(Y)$ over the whole collection of possible vertex covers $Y$.

**Proposition 2.27.** The complement of an OWA$_D$ solution for the MWIS problem is an OWA$_D$ solution for the MWVC problem, and vice-versa.

*Proof.* Let $X$ be any independent set and $Y$ its complement. Then $F(X,s) + \bar{F}(Y,s) = T_s$ and $F_s^* + \bar{F}_s^* = T_s$ for any scenario $s \in S$. Moreover, corresponding regrets must be equal:

$$F_s^* - F(X,s) = T_s - \bar{F}_s^* - T_s + \bar{F}(Y,s) = \bar{F}(X,s) - \bar{F}_s^*.$$

If their regrets are equal, then their descending order of regrets must be equal i.e. permutations $\psi$ from OWA$_D$ for vertex covers coincide with permutations $\sigma$ from OWA$_D$ for independent sets. Consequently, it holds:

$$\begin{aligned} O_D(X) &= \sum_{i=1}^{p} a_i \cdot (F^*_{\sigma(i)} - F(X, s_{\sigma(i)})) = \sum_{i=1}^{p} a_i \cdot (\bar{F}(Y, s_{\sigma(i)}) - \bar{F}^*_{\sigma(i)}) \\ &= \sum_{i=1}^{p} a_i \cdot (\bar{F}(Y, s_{\psi(i)}) - \bar{F}^*_{\psi(i)}) = \bar{O}_D(Y). \end{aligned}$$

Assume now that $X_{OD}$ is an OWA$_D$ solution for the MWIS problem, and that $Y_{OD}$ is the complement of $X_{OD}$. Then:

$$\begin{aligned} O_D(X) &\quad \text{achieves minimum for} \quad X = X_{OD} \quad \Longrightarrow \\ \bar{O}_A(Y) &\quad \text{achieves minimum for} \quad Y = Y_{OD}. \end{aligned}$$

Thus $Y_{OD}$ is by definition an OWA$_D$ solution for the MWVC problem. The proof in opposite direction is conducted analogously. ■

The third OWA-type criterion that can also be considered is the OWA$_R$ criterion, which is a generalization of traditional relative robustness. It is clear that the OWA$_R$ variant of the MWIS problem cannot in general be equivalent to the OWA$_R$ variant of the MWVC problem. As a counterexample, Example 2.14 from Section 2.3 can again be used. However, the sufficient condition for equivalence specified by Proposition 2.13 still holds, i.e. Proposition 2.13 can easily be generalized from the traditional relative min-max regret to the OWA$_R$ criterion.

Again, for a chosen independent set $X$ and vertex cover $Y$ respective relative regrets $(F_s^* - F(X,s))/F_s^*$ and $(\bar{F}(Y,s) - \bar{F}_s^*)/\bar{F}_s^*$ are sorted in descending order, i.e. permutations $\sigma$ and $\psi$ are found such that

$$\frac{F^*_{s_{\sigma(1)}} - F(X,s_{\sigma(1)})}{F^*_{s_{\sigma(1)}}} \geq \frac{F^*_{s_{\sigma(2)}} - F(X,s_{\sigma(2)})}{F^*_{s_{\sigma(2)}}} \geq \dots \geq \frac{F^*_{s_{\sigma(p)}} - F(X,s_{\sigma(p)})}{F^*_{s_{\sigma(p)}}}$$

$$\frac{\bar{F}(Y,s_{\psi(1)}) - \bar{F}^*_{s_{\psi(1)}}}{\bar{F}^*_{s_{\psi(1)}}} \geq \frac{\bar{F}(Y,s_{\psi(2)}) - \bar{F}^*_{s_{\psi(2)}}}{\bar{F}^*_{s_{\psi(2)}}} \geq \dots \geq \frac{\bar{F}(Y,s_{\psi(p)}) - \bar{F}^*_{s_{\psi(p)}}}{\bar{F}^*_{s_{\psi(p)}}}.$$

Then *OWA$_R$ cost* for $X$ and $Y$ are computed as:

$$O_R(X) = \sum_{i=1}^{p} a_i \cdot \frac{F^*_{s_{\sigma(i)}} - F(X,s_{\sigma(i)})}{F^*_{s_{\sigma(i)}}}$$

$$\bar{O}_R(Y) = \sum_{i=1}^{p} a_i \cdot \frac{\bar{F}(Y,s_{\psi(i)}) - \bar{F}^*_{s_{\psi(i)}}}{\bar{F}^*_{s_{\psi(i)}}}.$$

**Definition 2.28.** An OWA$_R$ solution for the MWIS problem is an independent set $X_{OR}$ that minimizes the function $O_R(X)$ over the whole collection of possible independent sets $X$.

**Definition 2.29.** An OWA$_R$ solution for the MWVC problem is a vertex cover $Y_{OR}$ that minimizes the function $\bar{O}_R(Y)$ over the whole collection of possible vertex covers.

**Proposition 2.30.** Suppose that the ratio among optimal solution weights for the conventional MWIS and MWVC problem, respectively, is the same for all scenarios. Or in other words, suppose that $\bar{F}_s^*/F_s^* = Q$ for all $s \in S$. Then the complement of an OWA$_R$ solution for the MWIS problem is an OWA$_R$ solution for the MWVC problem, and vice-versa.

*Proof.* Let $X$ be any independent set and $Y$ its complement. Then $F(X,s) + \bar{F}(Y,s) = T_s$ and $F_s^* + \bar{F}_s^* = T_s$ for any scenario $s \in S$. Moreover, corresponding relative regrets must be proportional:

$$\frac{F_s^* - F(X,s)}{F_s^*} = \frac{\bar{F}_s^*}{F_s^*} \cdot \frac{T_s - \bar{F}_s^* - T_s + \bar{F}(Y,s)}{\bar{F}_s^*} = Q \cdot \frac{\bar{F}(Y,s) - \bar{F}_s^*}{\bar{F}_s^*}.$$

If the corresponding relative regrets are proportional, then their descending order must be equal i.e. permutations $\psi$ from OWA$_R$ for vertex covers coincide with permutations $\sigma$ from OWA$_R$ for independent sets. Consequently, it holds:

$$\begin{aligned} O_R(X) &= \sum_{i=1}^{p} a_i \cdot \frac{F_{\sigma(i)}^* - F(X, s_{\sigma(i)})}{F_{\sigma(i)}^*} = Q \cdot \sum_{i=1}^{p} a_i \cdot \frac{\bar{F}(Y, s_{\sigma(i)}) - \bar{F}_{\sigma(i)}^*}{\bar{F}_{\sigma(i)}^*} \\ &= Q \cdot \sum_{i=1}^{p} a_i \cdot \frac{\bar{F}(Y, s_{\psi(i)}) - \bar{F}_{\psi(i)}^*}{\bar{F}_{\psi(i)}^*} = Q \cdot \bar{O}_R(Y). \end{aligned}$$

Assume now that $X_{OR}$ is an OWA$_R$ solution for the MWIS problem, and that $Y_{OR}$ is the complement of $X_{OR}$. Then:

$$\begin{aligned} O_R(X) &\text{ achieves minimum for } X = X_{OR} \implies \\ Q \cdot \bar{O}_R(Y) &\text{ achieves minimum for } Y = Y_{OR} \implies \\ \bar{O}_R(Y) &\text{ achieves minimum for } Y = Y_{OR}. \end{aligned}$$

Thus $Y_{OR}$ is by definition an OWA$_R$ solution for the MWVC problem. The proof in the opposite direction is conducted analogously.

■

## 2.6. Pareto efficiency

In this section we consider solving robust MWIS and MWVC problem by the method based on Pareto efficiency. Let us denote again the whole list of scenarios as $s_1, s_2, \ldots, s_p$. By applying the general ideas of dominance and efficiency from Chapter 1 to our problems, the following more concrete definitions are obtained.

**Definition 2.31.** An independent set $X$ is dominated by another independent set $\widetilde{X}$ if it holds that:

$$(F(X,s_1),F(X,s_2),\ldots,F(X,s_p)) < (F(\widetilde{X},s_1),F(\widetilde{X},s_2),\ldots,F(\widetilde{X},s_p)).$$

An independent set $X$ is efficient if it is not dominated by any other independent set.

**Definition 2.32.** A vertex cover $Y$ is dominated by another vertex cover $\widetilde{Y}$ if it holds that:

$$(\bar{F}(Y,s_1),\bar{F}(Y,s_2),\ldots,\bar{F}(Y,s_p)) > (\bar{F}(\widetilde{Y},s_1),\bar{F}(\widetilde{Y},s_2),\ldots,\bar{F}(\widetilde{Y},s_p)).$$

A vertex cover $Y$ is efficient if it is not dominated by any other vertex cover.

In the above definitions, vector notation has been used. Also, the vectors have been written as rows. The ordering of vectors is defined in the standard way, i.e. componentwise. Indeed, for two vectors $\vec{a} = (a_1,a_2,\ldots,a_p)$ and $\vec{b} = (b_1,b_2,\ldots,b_p)$ it holds that $\vec{a} \leq \vec{b}$ if $a_i \leq b_i$ for all $i = 1,2,\ldots,p$. Next, $\vec{a} < \vec{b}$ means that $\vec{a} \leq \vec{b}$ and $\vec{a} \neq \vec{b}$ .

The method for solving the robust MWIS problem analyzed in this section consists of finding the whole collection of efficient independent sets, i.e. those independent sets that are not dominated by some other independent set. The analogous method for solving the robust MWVC problem consists of finding the whole collection of efficient vertex covers.

As always before, we would like to know whether the proposed ways of solving the MWVIS and MWVC problem are equivalent in the sense that the solution of one problem can easily be transformed into the solution of the other problem. Once more, the answer is positive, as guaranteed by the following proposition.

**Proposition 2.33.** The collection of complements of all efficient independent sets coincides with the collection of all efficient vertex covers, and vice-versa.

*Proof.* Let $X$ be an efficient independent set. Let $Y$ be the complement of $X$. We claim that $Y$ must be an efficient vertex cover. Indeed, If $Y$ is not efficient, then there exists another vertex cover $\widetilde{Y}$ that dominates over $Y$. Denote with $\widetilde{X}$ the complement of $\widetilde{Y}$. Then it holds

$$(\bar{F}(Y,s_1),\bar{F}(Y,s_2),\ldots,\bar{F}(Y,s_p)) > (\bar{F}(\widetilde{Y},s_1),\bar{F}(\widetilde{Y},s_2),\ldots,\bar{F}(\widetilde{Y},s_p)),$$

or equivalently

$$(T_{s_1} - F(X,s_1),T_{s_2} - F(X,s_2),\ldots,T_{s_p} - F(X,s_p)) >$$
$$(T_{s_1} - F(\widetilde{X},s_1),T_{s_2} - F(\widetilde{X},s_2),\ldots,T_{s_p} - F(\widetilde{X},s_p)).$$

or componentwise

$$T_{s_i} - F(X, s_i) \geq T_{s_i} - F(\widetilde{X}, s_i), \ i = 1, 2, \ldots, p \ \text{(inequality is strict for at least one } i\text{)},$$

which is equivalent to

$$F(X, s_i) \leq F(\widetilde{X}, s_i), \ i = 1, 2, \ldots, p \ \text{(inequality is strict for at least one } i\text{)},$$

or in vector notation

$$(F(X, s_1), F(X, s_2), \ldots, F(X, s_p)) < (F(\widetilde{X}, s_1), F(\widetilde{X}, s_2), \ldots, F(\widetilde{X}, s_p)).$$

So $X$ is dominated by $\widetilde{X}$, which is a contradiction to the initial assumption that $X$ is efficient. Thus $Y$ must also be efficient.

So far we have proved that the collection of complements of all efficient independent sets must be a sub-collection of the collection of all efficient vertex covers. However, the same proof can be conducted in the opposite direction, thus proving that the collection of complements of all efficient vertex covers must be a sub-collection of the collection of all efficient independent sets. Thanks to such bi-directionality, it is obvious that the considered collections must coincide, i.e. the mentioned inclusions are in fact equalities. ■

According to the proposition above, any algorithm [9, 20] that determines all efficient independent sets could be used as an algorithm for determining all efficient vertex covers, and vice-versa.

In the remaining part of this section, we will analyze relationships among particular robust solutions from Sections 2.1-2.5 and efficient solutions. Such relationships are summarized by the following two propositions.

**Proposition 2.34.** An absolute robust solution for the MWIS problem can be chosen so that it is also an efficient solution for the MWIS problem. The same claim is also true regarding a (relative) robust deviation solution or OWA solutions for the MWIS problem. The analogous claims are valid for the MWVC problem as well.

*Proof.* Let $X_{OA}$ be an OWA$_A$ solution for the MWIS problem. If $X_{OA}$ is not an efficient independent set, then there exists another independent set $\widetilde{X}$ that is efficient and that dominates over $X_{OA}$. Thus:

$$F(X_{OA}, s_i) \leq F(\widetilde{X}, s_i), \quad i = 1, ..., p. \tag{2.1}$$

Let denote $\sigma$ and $\psi$ permutations which sort arrays of weights for $X_{OA}$ and for $\tilde{X}$ in ascending order, respectively. Then:

$$F(X_{OA}, s_{\sigma(i)}) \leq F(\widetilde{X}, s_{\psi(i)}), \quad i = 1, ..., p. \tag{2.2}$$

Hence, after sorting both of arrays ascending, any element within the array for $X_{OA}$ is less or equal to corresponding element within the array for $\widetilde{X}$. In order to prove statement (2.2), let sort both of arrays by permutation $\psi$. From (2.1) follows:

$$F(X_{OA}, s_{\psi(i)}) \leq F(\widetilde{X}, s_{\psi(i)}), \quad i = 1, ..., p.$$

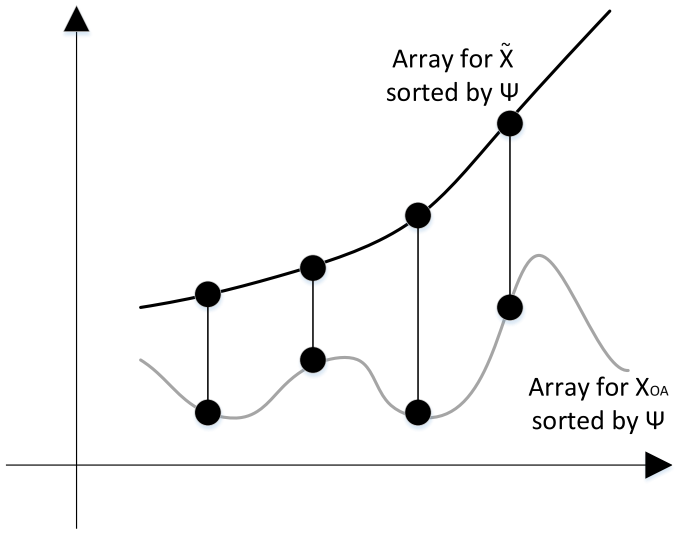

Figure 2.5: Arrays of weights for $X_{OA}$ and $\tilde{X}$ sorted by permutation $\psi$. Elements of $X_{OA}$ and $\tilde{X}$ are shown on gray and black line, respectively.

In Figure 2.5 we see both arrays sorted by permutation $\psi$. Elements of $\tilde{X}$ are sorted ascending. Further, elements of $X_{OA}$ do not have to be sorted by any specific rule, but they are always under the corresponding elements of $\tilde{X}$.

Now we sort the array for $X_{OA}$ into an ascending order. Sorting into ascending order will not corrupt the fact that all elements of $X_{OA}$ are under some elements of array for $\tilde{X}$. Indeed, let us use the selection sort for rearranging elements of $X_{OA}$. Each step of the selection sort swaps currently the smallest element with the element at the leftmost position. After swapping, the smallest element would be under the black line because it is smaller than the previous element from that position. Also, the element that previously was at the leftmost position moves to the right, and therefore it must remain under the (ascending) black line when it arrives at its new position. Hence, after resorting elements of $X_{OA}$ we have a following layout of elements:

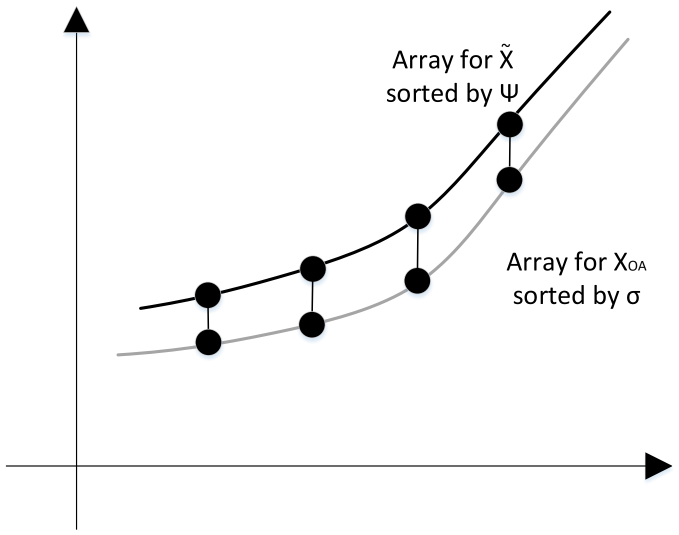

Figure 2.6: Arrays of weights for $X_{OA}$ and $\tilde{X}$ sorted by permutation $\psi$ and $\sigma$, respectivelly. All elements of $\tilde{X}$ are smaller then corresponding elements of $X_{OA}$, e.g. the gray line in always under the black one.

This proves the statement (2.2). Furthermore:

$$a_i \cdot F(X_{OA}, s_{\sigma(i)}) \leq a_i \cdot F(\widetilde{X}, s_{\psi(i)}), \quad i = 1, 2, ..., p.$$

From which follows:

$$\sum_{i=1}^{p} a_i \cdot F(X_{OA}, s_{\sigma(i)}) \leq \sum_{i=1}^{p} a_i \cdot \bar{F}(\tilde{X}, s_{\psi(i)}),$$

or in other words:

$$O_A(X_{OA}) \leq O_A(\tilde{X}). \tag{2.3}$$

By assumption, $X_{OA}$ is an OWA$_A$ solution i.e. it maximizes $O_A$. Because of (2.3), $\tilde{X}$ also maximizes $O_A$ i.e. it is also an OWA$_A$ solution. Hence we have found an OWA$_A$ solution which is also an efficient solution..

The claims stated for OWA$_D$ and OWA$_R$ are proved analogously. The absolute robustness and the (relative) robust deviation are special cases of OWA. All claims for MWVC also follow analogously.

■

**Proposition 2.35.** Suppose that all coefficients $s_1, s_2, \dots, s_p$ within an OWA criterion are nonzero. Then any OWA solution for the MWIS problem must be an efficient solution for the MWIS problem. An analogous claim is also valid for the MWVC problem.

*Proof.* Let $X_{OA}$ be an OWA$_A$ solution for the MWIS problem. If $X_{OA}$ is not an efficient independent set, then there exists another independent set $\widetilde{X}$ that is efficient and that dominates over $X_{OA}$. Thus:

$$F(X_{OA}, s_i) \leq F(\widetilde{X}, s_i) \quad i = 1, ..., p, \tag{2.4}$$

and at least one inequality is strict.

Let denote $\sigma$ and $\psi$ permutations which sort arrays of weights for $F(X, s_i)$ and for $F(\tilde{X}, s_i)$ in ascending order, respectively. In the previous Proposition 3.33 we have proved:

$$F(X_{OA}, s_{\sigma(i)}) \leq F(\widetilde{X}, s_{\psi(i)}) \quad i = 1, ..., p. \tag{2.5}$$

Moreover, there exist at least one $j \in \{1, ..., p\}$ such that:

$$F(X_{OA}, s_{\sigma(j)}) < F(\widetilde{X}, s_{\psi(j)}).$$

Namely, if such $j$ did not exist, then all inequalities (2.5) would actually be equalities and the sums of weights would be equal i.e.

$$\sum_{i=1}^{p} F(X_{OA}, s_i) = \sum_{i=1}^{p} F(\tilde{X}, s_i). \tag{2.6}$$

But, on the other hand, (2.6) cannot hold because of (2.4), where at least one strict inequality exists. Next, because $a_i \neq 0$, $i = 1, ..., p$ it must be true that:

$$a_j \cdot F(X_{OA}, s_{\sigma(j)}) < a_j \cdot F(\widetilde{X}, s_{\psi(j)})$$
$$a_i \cdot F(X_{OA}, s_{\sigma(i)}) \leq a_i \cdot F(\widetilde{X}, s_{\psi(i)}), \quad i \neq j.$$

When we sum up the above inequalities we get:

$$\sum_{i=1}^{p} a_i \cdot F(X, s_{\sigma(i)}) < \sum_{i=1}^{p} a_i \cdot F(\tilde{X}, s_{\psi(i)}),$$

or in other words:

$$O_A(X_{OA}) < O_A(\tilde{X}). \tag{2.7}$$

But this is in contradiction with our assumption that $X_{OA}$ is an OWA$_A$ solution thus maximizing $O_A()$. Hence, $X_{OA}$ must be an efficient solution.

The claims stated for OWA$_D$ or OWA$_R$ or the MWVC problem are proved analogously.

■

The following two corollaries are obtained as simple consequences of Proposition 2.33 combined with Proposition 2.34 and 2.35, respectively.

**Corollary 2.36.** An absolute robust solution for the MWIS problem can be chosen so that its complement is an efficient solution for the MWVC problem. Also, an absolute robust solution for the MWVC problem can be chosen so that its complement is an efficient solution for the MWIS problem. The same claims are also true regarding (relative) robust deviation solutions or OWA solutions.

**Corollary 2.37.** Suppose that all coefficients $s_1, s_2, \ldots, s_p$ within an OWA criterion are nonzero. Then the complement of any OWA solution for the MWIS problem must be an efficient solution for the MWVC problem. Also, the complement of any OWA solution for the MWVC problem must be an efficient solution for the MWIS problem.

Roughly speaking, Corollary 2.36, and specially Corollary 2.37, say the following. Although the complement of a robust solution for the MWIS problem may not necessarily be an equivalent robust solution for the MWVC problem, it should still be an efficient solution for the MWVC problem. Analogous interpretation the other way around is also valid.

Now we give one more example which illustrates Propositions 2.33 and 2.34 as well as Corollary 2.36.

**Example 2.38.** Let us again consider the graph from Figure 2.2. Then the corresponding efficient solutions of the MWIS and the MWVC problem are are presented in Table 2.6. The data shown in Table 2.6 are the same as in Table 2.3, only the highlighting is different, i.e. now all efficient independent sets and vertex covers are printed in boldface.

As we can see, there are two efficient independent sets, consisting of vertices 0,2,3,6,8 and 0,2,5,8, respectively. There are two efficient vertex covers as well, comprising vertices 1,4,5,7 and 1,3,4,6,7, respectively. According to Proposition 2.33, complements of efficient independent sets are efficient vertex covers, and vice-versa. By comparison with Table 2.3 and 2.4, we can check that all previously found robustly optimal solutions of any kind are at the same time efficient solutions, which is in accordance with Proposition 2.34. Finally, the complement of any robustly optimal solution is at least efficient (if not again robustly optimal), which is in accordance with Corollary 2.36.

Table 2.6: Finding efficient solutions for the graph shown in Figure 2.2

| Independent set | Weight for each scenario | | | Vertex cover | Weight for each scenario | | |
|---|---|---|---|---|---|---|---|
| 0,2,3,6,7 | 13 | 14 | 16 | 1,4,5,8 | 32 | 31 | 29 |
| **0,2,3,6,8** | 34 | 26 | 33 | **1,4,5,7** | 11 | 19 | 12 |
| 0,2,4,6,7 | 14 | 17 | 17 | 1,3,5,8 | 31 | 28 | 28 |
| 0,2,5,7 | 11 | 16 | 14 | 1,3,4,6,8 | 34 | 29 | 31 |
| **0,2,5,8** | 32 | 28 | 31 | **1,3,4,6,7** | 13 | 17 | 14 |
| 1,4,6,7 | 10 | 15 | 12 | 0,2,3,5,8 | 35 | 30 | 33 |
| 1,5,7 | 7 | 14 | 9 | 0,2,3,4,6,8 | 38 | 31 | 36 |
| 1,5,8 | 28 | 26 | 26 | 0,2,3,4,6,7 | 17 | 19 | 19 |

# 3. An evolutionary algorithm for robust variants of the MWIS problem

Having in mind the mentioned NP-hardness of the conventional MWIS problem, in this chapter we propose an approximate algorithm for solving robust variants of the MWIS problem based on evolutionary computing. More precisely, the algorithm will be used for solving problem variants based on three robustness criteria: absolute robustness, robust deviation and relative robust deviation.

## 3.1. Basic properties of our EA

As described in many books, e.g. [10, 22, 29], an evolutionary algorithm (EA) is a randomized computing procedure which maintains a population of *chromosomes*. Each chromosome represents a feasible solution to a given instance of an optimization problem. The population is iteratively changed, thus giving a series of population versions called *generations*. It is expected that the best chromosome in the last generation should represent a nearly optimal solution to the considered problem instance.

An EA consists of many building blocks, which can be chosen and combined in different ways. Consequently, there are many possible EA variants for the same optimization problem. Some important building blocks are:

- Data structure used to represent a chromosome.
- Evaluation procedure used to assess "goodness" or "fitness" of a chromosome.

- Crossover operators, which create new chromosomes (children) by combining parts of several (usually two) existing chromosomes (parents).
- Recovery operators, which modify chromosomes that correspond to infeasible solutions in order to make them feasible.
- Mutation operators, which make a small and apparently random change in a single existing chromosome (mutant).
- Selection procedure used to find "good" chromosomes for crossover, or "bad" chromosomes that will be discarded from the population.
- Insertion procedure, which inserts newly produced chromosomes (children) into the current population, while keeping the total population size under control.

In order to construct a good EA for the RMWIS problem, we will consider four original crossover operators and five mutation operators. Also, we will propose a flexible recovery operator. In this way, altogether twenty EA variants will be considered, where any of them combines one particular crossover with one particular mutation. We will apply each EA variant to each of the three mentioned robustness criteria.

The mentioned operators (crossovers, mutations, recovery) will be described in more detail in subsequent sections. The remaining building blocks of our EA are realized in standard ways and they do not require extensive elaboration. Indeed:

- The data structure used for a chromosome is essentially a list of vertices, which can be sorted according to various criteria.
- The evaluation procedure simply computes the objective function value according to the chosen criterion of robustness.
- Selection of a "good" (or "bad") chromosome is based on the well-known tournament selection [10,29]. Thus a certain number of chromosomes is picked up randomly, and the most fit (or the least fit) of them is chosen.
- The insertion procedure for children is the same as described in [26], and it relies on the concept of similarity. We say that two chromosomes are *similar* if their robust objective-function values differ less than 1% of the best value within the population. Insertion is accomplished according to the following two rules. If there exists another chromosome

in the population that is similar to the new one, then the better of those two "twins" is retained and the other one is discarded. If there is no similar chromosome, then the new one is retained and some other "bad" chromosome is selected by tournament and discarded.

Note that according to our rules the population size always remains the same as it was at the beginning. Indeed, whenever a chromosome is inserted, another one is discarded. Note also that the whole algorithm supports the so-called *elitism* [10,29], i.e. the best chromosome within the population is discarded only if it is replaced by an even better chromosome.

## 3.2. The greedy vertex-selecting rule

Since an EA usually requires many iterations, it is important that a particular iteration can be computed quickly. Therefore, in our crossover and recovery operators, we will mostly rely on greedy strategies. It is true that greedy decisions often lead to an unsatisfactory local optimum. However, such premature termination of computing can be avoided by efficient mutation operators.

Our greedy strategies are mostly inspired by the vertex-selecting rule from [28], which maximizes:

$$c(v_i) = \frac{w_i}{d(v_i)+1}. \tag{3.1}$$

Here, $w_i$ is the weight of vertex $v_i$, and $d(v_i)$ is its degree (the number of adjacent vertices). The justification for the expression above is illustrated by Figure 3.1. Indeed, for the shown graph and the shown weights of vertices, a vertex-selecting rule which only took weights into consideration would give an independent set of total weight 7. On the other hand, the vertex-selecting rule based on (3.1) would give an independent set of total weight 8. Sometimes vertices with larger weights can have many neighbors, so that better solutions can be obtained by using a larger number of more independent vertices with smaller weights.

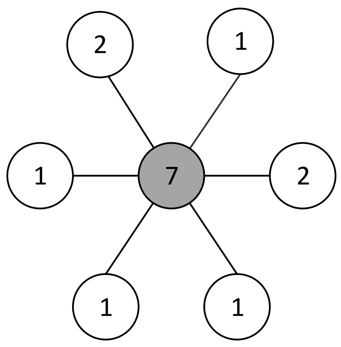

Figure 3.1: A weighted graph illustrating various vertex-selecting rules

The expression (3.1) can be generalized for the robust case as follows:

$$c(v_i) = \frac{\sum_{s \in S} w_i^s}{d(v_i) + 1}, \tag{3.2}$$

where $w_i^s$ is the weight of a vertex $v_i$ under scenario $s$. Thus according to our vertex-selecting rule, we give priority to a vertex $v_i$ whose ratio (3.2) is the greatest (provided that it is not adjacent to other already chosen vertices). The ratio (3.2) we will be called *relative vertex contribution*, and its numerator will be referred to as *total weight*.

## 3.3. Crossover operators

Now we will describe our four original crossover operators that seem to be suitable for solving the robust variants of the problem. As already mentioned, each crossover takes as input two existing chromosomes (parents) and produces a new chromosome (child). Thereby both parents correspond to feasible solutions of the considered RMWIS problem instance. On the other hand, the child may for some crossovers turn out to be infeasible, which is then compensated by applying recovery. Each chromosome is represented as a list of chosen vertices, and that list can be sorted in various ways.

- *Alternating vertices crossover (AVX).* Our first crossover operator, called AVX, assumes that within each parent list the vertices are sorted according to their relative vertex contributions (3.2) in a descending order. The child chromosome is constructed by choosing in alternation vertices from the first and from the second parent, thereby following the ordering in both lists. Thus the child is initialized with the first vertex from the first parent, then the first vertex from the second parent is added, then the second vertices from the first and second parent are in turn appended, etc. In case of infeasibility (the candidate vertex is identical or adjacent to some of already chosen vertices) the next vertex from the same parent list is considered. If one of the parent lists gets exhausted, the remaining vertices from the other parent list are directly copied into the child (provided that they are feasible).

Although the child constructed in the above way is always feasible, it is still handed over to the recovery operator for possible improvement.

- *Modified alternating vertices crossover (MAVX).* The next operator, denoted with MAVX, is almost the same as AVX. The only difference is the way the parent lists are initially sorted. In AVX, the parents are sorted according to the relative vertex contribution (3.2) in a descending order. In MAVX, they are sorted according to the *local* relative vertex contribution, again in a descending order. Th local relative vertex contribution of a vertex $v_i$ is computed by the same formula (3.2), except that the degree $d(v_i)$ is now assessed "locally", i.e. by only considering edges connecting $v_i$ with vertices from the other parent.

- *Randomly chosen vertices crossover (RVX).* In RVX the parent lists do not need to be sorted in any particular order, but it is convenient to assume that they are sorted according to vertex identifiers $1, \ldots, n$. The child chromosome is constructed in $n$ steps. In $i$-th step it is decided whether to include vertex $v_i$ into the child or not. The decision is guided by the situation found in one of the parents, i.e. if $v_i$ is present in the chosen parent, then it will also be present in the child, and vice-versa. In each step, choosing between the two parents is done randomly, but not with equal probability. More precisely, let the total relative vertex contribution (the sum of relative contributions of included vertices) for the first and for the second parent be $t_1$ and $t_2$, respectively. Then the probability of choosing the first parent is
$$\pi_1 = \frac{t_1}{t_1 + t_2}$$
and the probability for the second parent is $\pi_2 = 1 - \pi_1$. Thus the parent with larger total relative vertex contribution, being considered as more fit, is more likely to be chosen.

  Obviously, the child chromosome created by the above procedure may not be feasible, so that it is always sent to the recovery operator for postprocessing.

- *Modified randomly chosen vertices crossover (MRVX).* The last crossover operator, MRVX, works almost in the same way as RVX. It differs only in the formula for computing the probability $\pi_1$. Instead of relative vertex contributions, now the robust objective function from the robust versions of MWIS problem is taken into account. Depending on the robust criteria (i.e. max-min, min-max-regret, or relative min-max regret), one of the following three formulas is used:

$$\begin{aligned}
\pi_1 &= \frac{F_A(\text{first parent})}{F_A(\text{first parent}) + F_A(\text{second parent})}, \text{ or} \\
\pi_1 &= \frac{F_D(\text{second parent})}{F_D(\text{first parent}) + F_D(\text{second parent})}, \text{ or} \\
\pi_1 &= \frac{F_R(\text{second parent})}{F_R(\text{first parent}) + F_R(\text{second parent})}.
\end{aligned}$$

Note that the max-min variant is a maximization problem, where larger values are regarded as better, while the other two variants are minimization problems, where smaller values are better. Consequently, the above three formulas are adjusted so that, for each robustness criterion, a more fit parent is more likely to be chosen.

At the end of this section we give two examples to illustrate how AVX and RVX crossovers work.

**Example 3.1.** The AVX operator is illustrated by Figure3.2. The same graph is drawn several times in order to show each parent separately, their child, and the recovered version of that child (if the recovery is needed). The vertices are labeled (in fact identified) according to their relative vertex contributions (3.2) sorted in a descending order, i.e. vertex 0 has the highest and vertex 8 the lowest contribution.

Let us now analyze in detail the example from Figure 3.2. The first parent is shown in light gray color, i.e. it consists of vertices 0, 2, 3, 6 and 8. The second parent is shown in black, i.e. it comprises vertices 1, 5 and 8. We can see that both parents are feasible. The child obtained by AVX combines vertices 0 and 2 from the first parent with vertices 5 and 8 from the second parent, which is indicated by colors. As explained earlier, AVX works in steps. In the first step the child is initialized with the light gray vertex 0. In the second step the black vertex 5 is inserted into the child after skipping the infeasible black vertex 1. In the third step the light gray vertex 2 is appended. In the fourth step the black vertex 8 is chosen. After that, both parent lists become exhausted (since the remaining light gray vertices are infeasible) and the procedure is finished. The obtained child is feasible by construction, but unfortunately it cannot be improved by adding more vertices.

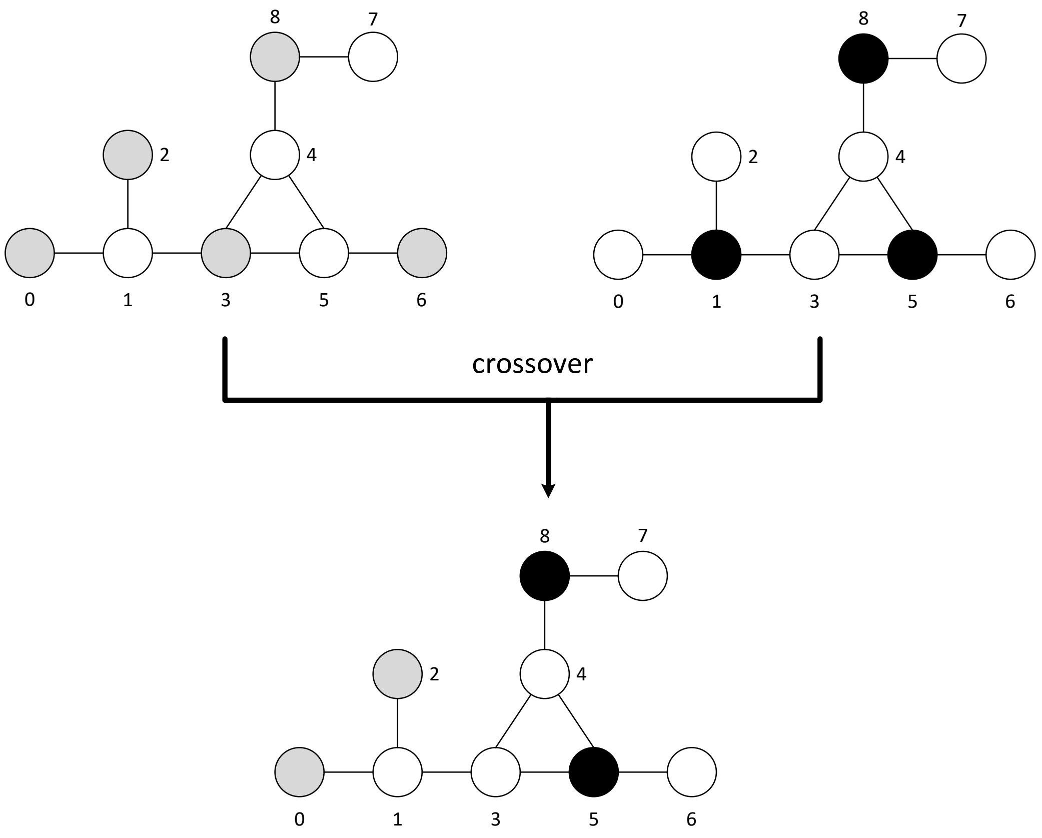

Figure 3.2: An example illustrating the AVX operator

**Example 3.2.** The RVX operator with recovery is illustrated by Figure 3.3. Again, the first and the second parent are shown by light gray and red black, respectively, and the vertices are labeled to their relative contribution (3.2). We see that the first parent contains vertices 0, 2, 3, 6 and 8, while the second parent consists of vertices 1, 4, 6 and 7. The child obtained by RVX in its recovered form comprises vertices 0, 2 and 6 from the first parent, vertex 4 from the second parent, and the reintroduced vertex 7 shown in dark gray. As explained before, RVX works in steps. During its step *i*, RVX decides whether to include vertex *i* into the child chromosome or not. The decision is guided by the situation found in the parent that has randomly been chosen within that step. In our example, the light gray parent is more likely to be chosen since its total relative contribution is larger. Suppose that the random choices made in steps 1,2, . . . , 9 are in turn: light gray, light gray, light gray, black, black, black, light gray, light gray and light gray, respectively. Then the initially obtained chromosome turns out to be infeasible since its chosen vertices 4 and 8 are adjacent. The final child chromosome is obtained by applying the recovery operator from the next section. As we can see, the recovery operator first removes the light gray vertex 8 to assure feasibility and then adds the dark gray vertex 7 for improvement.

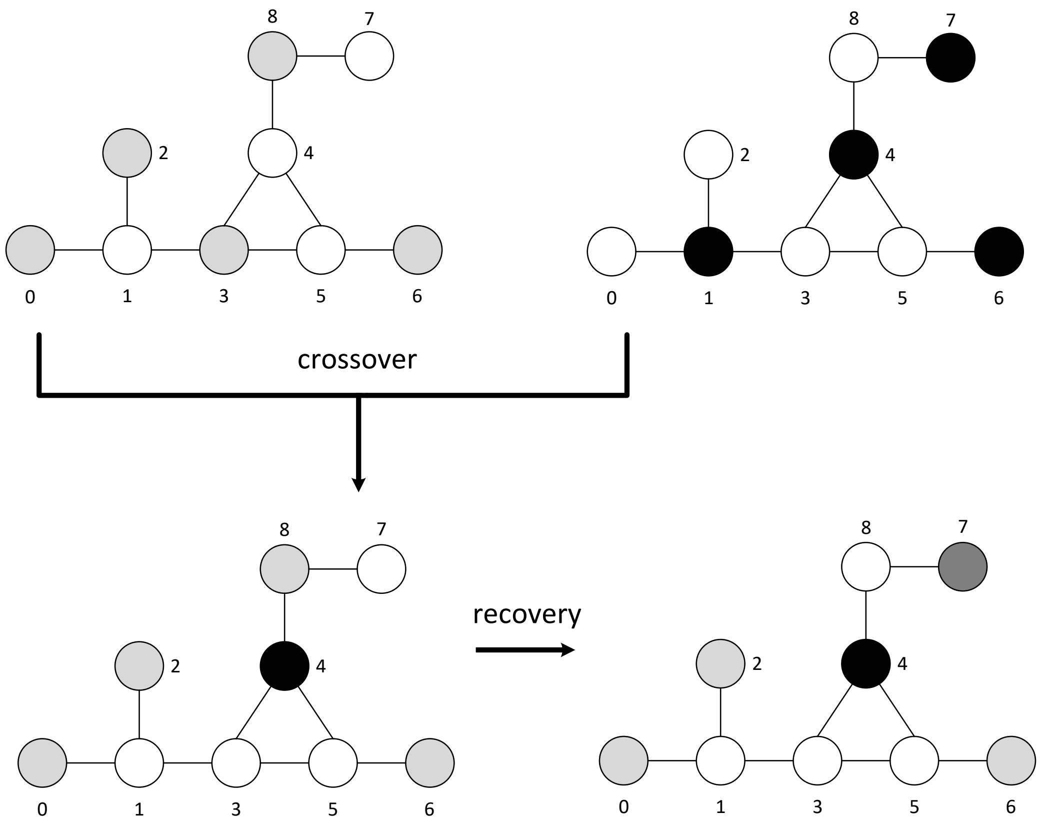

Figure 3.3: An example illustrating the RVX operator with recovery

## 3.4. Recovery operators

Our recovery operator takes as input a chromosome that may or may not correspond to a feasible solution of a considered robust versions of the MWIS problem. The operator produces a modified version of the same chromosome, which is in the first place feasible, but hopefully also more fit than the original version. The whole computing procedure consists of two phases:

1. Check if the given chromosome is feasible. If not, modify it so that it becomes feasible.
2. Try to improve the chromosome obtained in the first phase by further modification.

In the currently implemented variant of the operator, a chromosome is represented as a list of vertices that are sorted according to their total weights in descending order. The first phase of our procedure (assuring feasibility) is accomplished by rewriting the vertices from the original into a new (possibly shorter) list. Vertices are rewritten one by one, by following the original ordering. Thereby, any vertex may be skipped from rewriting if it would spoil feasibility of the already formed part of the new list.

The second phase of computing (improvement) is accomplished by inserting more vertices into the list obtained in the first phase. Thereby, only vertices from the given graph that have not been used in the original chromosome are considered. As we want to maintain genetic diversity, we tend to combine different vertex selecting rules. Therefore, for the second phase of recovery the candidates for insertion are processed in random order. Again, any of them is skipped if its insertion would spoil feasibility of the current list.

A concrete example of recovery has already been shown within Figure 3.3. Note that in Figure 3.3 both phases of recovery mentioned above (assuring feasibility, improvement) are applied. An additional example comprising only improvement will be given by Figure 3.6.

The described recovery operator allows many variants that have not been implemented at this moment, but could easily be tested in the future. For instance, the initial ordering of vertices could be different: instead of total weight we could use some other sorting criterion, e.g. the relative vertex contribution (3.2). Also, in the improvement phase, the vertices to be inserted can be processed in some other order, e.g. according to their total weights.

## 3.5. Mutation operators

Generally speaking, mutation is important because it maintains diversity of chromosomes from one generation to another. It also helps that the whole evolutionary process does not get stuck too early in a local optimum. In case of robust versions of the MWIS problem, the role of mutation would be to introduce new vertices, which exist in the given graph but are not used by current chromosomes.

As already mentioned before, each mutation takes one existing chromosome and modifies it through a small and apparently random change. In our case, a chromosome is in fact a list of chosen vertices. A mutation should make a change of that list by removing some vertices and inserting some other vertices. We will describe five mutation operators that work along that line.

- *Simple replacement mutation (SRM).* The SRM operator replaces a randomly chosen vertex in the original chromosome with a new vertex chosen randomly among those in the graph that have not been used by the original chromosome. The modified chromosome obtained in this way does not need to be feasible, so that it must further be processed by the recovery operator.

- *Weight-increasing replacement mutation (WIRM).* The WIRM operator is similar to SRM. But now the idea is that the overall procedure should be guided or at least influenced by a chosen scenario. Indeed, an old (existing) vertex within the chromosome is again replaced by a new (currently unused) vertex, and feasibility is again reestablished by applying the recovery operator. However, the choice of the two vertices is not completely random. Instead, care is taken that the new vertex has a larger weight than the old one according to the chosen scenario.

- *Weight-decreasing replacement mutation (WDRM).* The WDRM operator is similar to WIRM. But this time we replace the chosen vertex with a vertex having a smaller weight. Although WDRM spoils solutions, it also introduces new vertices which would otherwise be ignored.

- *Local search replacement mutation (LSRM).* For each scenario we make a new better chromosome (if possible) by improving the worst vertex in the original chromosome. The worst vertex $v$ is a vertex with the smallest weight. If there is a neighbor $\bar{v}$ of $v$ that has a larger weight than $v$ and is not adjacent to the other vertices in the original chromosome, we replace $v$ with $\bar{v}$. If there are more such vertices $\bar{v}$, we replace $v$ with the one among them having the largest weight. Finally, among all new chromosomes obtained for different scenarios, we choose the one with the best objective-function value ($F_A$, $F_D$ or $F_R$). Although the constructed chromosome is always feasible, it is still handed over to the recovery operator for possible improvement.

- *Complementary mutation (CM).* The CM operator can be regarded as an extreme variant of mutation, where a large number of new vertices is introduced simultaneously. For a given original chromosome, CM first determines the set $X \subseteq V$ of vertices in the graph $G$ that are used by that chromosome. Next, the new chromosome is made of vertices from $V \setminus X$ by using the greedy vertex selecting rule according to (3.2). More precisely, the vertices from $V \setminus X$ are sorted according to their relative vertex contribution in a descending order. Then, they are inserted in turn into the new chromosome. Again, any of those insertions are skipped if it would spoil the feasibility of the partially formed chromosome. Although the constructed chromosome is always feasible, it is still handed over to the recovery operator for possible improvement.

Finally, at the end of the section we give three examples to illustrate how SRM, LSRM and

CM mutations work.

**Example 3.3.** The SRM operator is illustrated by Figure 3.4. The original chromosome consists of vertices 1, 6 and 8, shown in light gray color on the left-hand side of the figure. The mutated chromosome is shown in a similar manner on the right-hand side, and it comprises vertices 1, 5 and 8. Thereby the original vertices are still light gray, and the newly introduced vertex is black. So we see that SRM has replaced vertex 6 with 5. The obtained chromosome is already feasible, and it cannot further be improved.

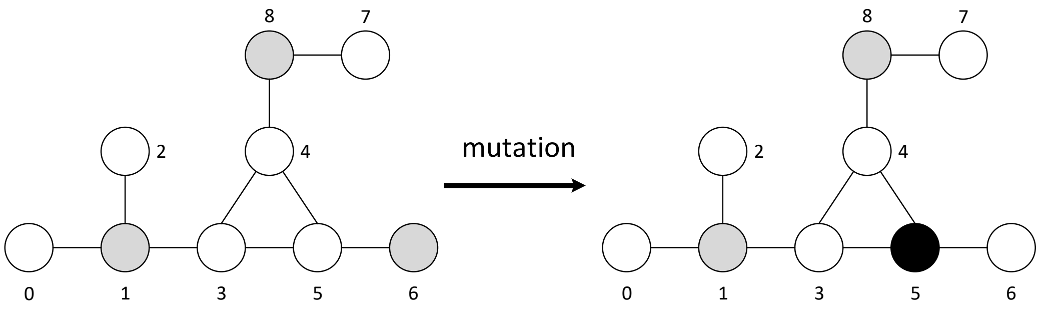

Figure 3.4: An example illustrating the SRM operator

**Example 3.4.** The LSRM operator is illustrated by Figure 3.5. For the sake of simplicity, let us have only one scenario. Vertices are again labeled according to their relative vertex contribution sorted in descending order, i.e. vertex 0 has the highest and vertex 8 the lowest contribution. The original chromosome, shown in light gray, consists of vertices 0, 2, 3, 6 and 8. Mutation picks vertex 8 for local improvement. Although, the best neighbor is vertex 4, replacing vertex 8 with vertex 4 would give an infeasible solution. Hence, vertex 8 is replaced with vertex 7, which is still better then vertex 8. The constructed chromosome is feasible and no further improvements are possible. In case of more scenarios, the above procedure would be made for the each scenario and then the best chromosome (according to the chosen objective-function) would be chosen.

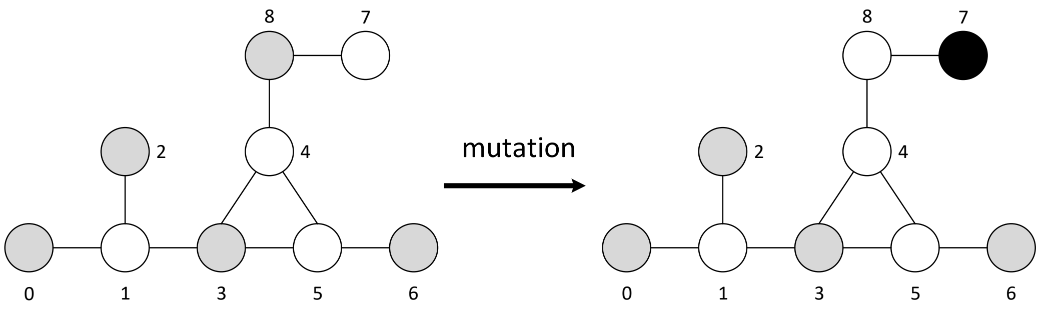

Figure 3.5: An example illustrating the LSRM operator

**Example 3.5.** The CM operator is illustrated by Figure 3.6. The original chromosome is again shown on the left-hand side of the figure - it consists of vertices 0, 2, 3, 6 and 8 colored light gray. The CM operator constructs the mutated chromosome from the complementary vertices 1, 4, 5 and 7, by using the greedy vertex selecting rule (3.2) as explained before. The result is shown on the right-hand side of the figure - it comprises vertices 1, 4 and 7 which are painted black. Note that the complementary vertex 5 has been skipped since it is adjacent to 4. The obtained chromosome is feasible, but it is still handed over to the recovery operator for improvement. The recovery operator improves the chromosome by adding vertex 6 painted dark gray in the lower part of the figure.

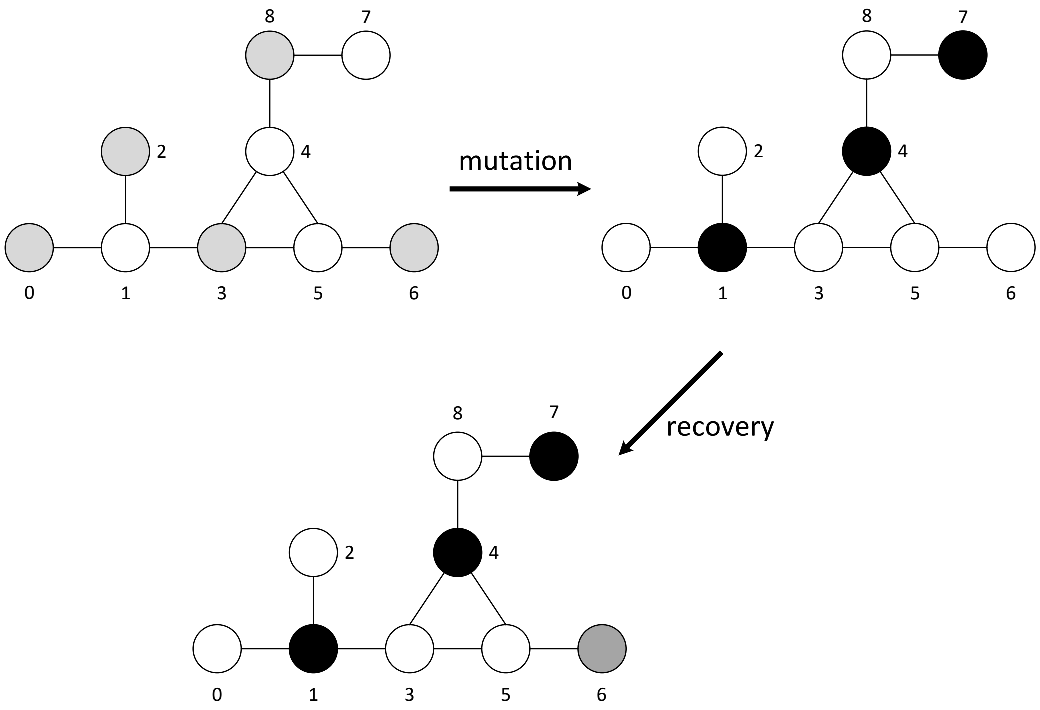

Figure 3.6: An example illustrating the CM operator

## 3.6. Linear programming formulation of the MWIS problem

Before we begin testing the previously mentioned crossover and mutation operators, it is important to note that the conventional MWIS problem can be written formally as an integer linear programming problem, as shown below. Such linear programming definition is necessary in order to be able to solve smaller problem instances by general-purpose optimization packages, e.g. by IBM ILOG CPLEX Optimization Studio [14].

$$\sum_{i=1}^{n} w_i x_i \to \max$$

subject to:

$$x_i + x_j \leq 1, \text{for all } (v_i, v_j) \in E$$

$$x_i \in \{0,1\}, \text{for all } i = 1, \ldots, n.$$

Here, $n$ is again the number of vertices in our graph $G$, i.e. $n = |V|$. The symbols $v_i$, $i = 1, \ldots, n$, denote vertices, and $w_i$, $i = 1, \ldots, n$, are their weights. The decision variable $x_i$ corresponds to $v_i$ and it equals 1 if and only if $v_i$ is included in the solution. The first condition assures that no two vertices within the solution are adjacent. We maximize the sum of weights among all feasible solutions. So the objective function can also be written as:

$$F(X) = \sum_{v_i \in X} w_i,$$

where $X$ is the set of chosen vertices, i.e. those vertices $v_i$ whose corresponding values $x_i$ are 1. In the robust case, the objective function at scenario $s$ is the following:

$$F(X,s) = \sum_{v_i \in X} w_i^s,$$

Here $w_i^s$ denotes the weight of vertex $v_i$ under scenario $s$.

Similarly as the conventional MWIS problem, our three basic variants of the robust MWIS problem can also be formulated in terms of linear programming and hopefully solved by general-purpose optimization packages. Indeed:

- *Absolute robust MWIS problem (max-min)*

$$
\begin{aligned}
& opt_A = y \to \max \\
& \text{subject to:} \\
& \sum_{i=1}^{n} w_i^s x_i \geq y, \text{for all } s \in S \\
& x_i + x_j \leq 1, \text{for all } (v_i, v_j) \in E \\
& x_i \in \{0,1\}, \text{for all } i = 1,...,n.
\end{aligned}
$$

- *Robust deviation MWIS problem (min-max regret)*

$$
\begin{aligned}
& opt_D = y \to \min \\
& \text{subject to:} \\
& F_s^* - \sum_{i=1}^{n} w_i^s x_i \leq y, \text{for all } s \in S \\
& F_s^* \text{is the optimal solution value for the conventional problem under scenario } s \\
& x_i + x_j \leq 1, \text{for all } (v_i, v_j) \in E \\
& x_i \in \{0,1\}, \text{for all } i = 1,...,n,
\end{aligned}
$$

- *Relative robust deviation MWIS problem (relative min-max regret)*

$$
\begin{aligned}
& opt_R = y \to \min \\
& \text{subject to:} \\
& \sum_{i=1}^{n} w_i^s x_i \geq (1-y) F_s^*, \text{for all } s \in S \\
& F_s^* \text{is the optimal solution value for the conventional problem under scenario } s \\
& x_i + x_j \leq 1, \text{for all } (v_i, v_j) \in E \\
& x_i \in \{0,1\}, \text{for all } i = 1,...,n,
\end{aligned}
$$

As far as the OWA criteria are concerned, they can be also formulated in terms of linear programming. However, such formulations, although correct from the theoretical point of view, turn out to be too demanding for practical computation. Therefore we will omit OWA criteria from our testing.

## 3.7. Testing and results

As there are (to best of our knowledge) no benchmarks for robust variants of the MWIS problem, we have generated our own two test groups, each comprising 30 problem instances based on random graphs. Any graph from the first group consists of 300 vertices, 30000 edges and 10 scenarios for vertex weights. A graph from the second group has the same number of vertices and scenarios, but only 20000 edges. Generally speaking, problem instances from the second group are harder to solve since their graphs are sparser, thus allowing more independent sets. In all instances and all scenarios, vertex weights range from 1 to 300. Indeed, we did not want to have vertices with extremely large weights since they would make optimal solutions too predictable and too easy to find. A full specification of all problem instances can be found in Appendix B of this thesis.

The chosen problem instances from our two test groups can be regarded as large enough to be nontrivial, but still small enough to be solved exactly by a general-purpose optimization package. Indeed, all instances (written in the form of integer linear programming) can be solved to optimality by the IMB ILOG CPLEX Optimization studio [14]. However, slightly more complex instances, e.g. those with 300 vertices but 10000 or 15000 edges, cannot be processed by CPLEX anymore (at least not on our computer) since they produce out-of-memory errors. Moreover, such slightly more complex instances cannot even be solved approximately by relaxation, since the obtained relaxed solutions usually consist of decision variables $x_i$ equal to 0.5, i.e. variables that cannot easily be rounded to a feasible set of 0-s and 1-s.

Our EA for solving the RMWIS problem has been implemented in the C++ programming language by using the Microsoft Visual Studio programming package [23]. The implementation supports all previously mentioned EA variants obtained by combining one among four proposed crossover operators with one among five mutation operators. The program source code can again be found in Appendix A of this dissertation.

The implemented algorithm has been tested on a computer with an Intel Core i5-6600K @ 3.50 GHz processor and 16 GB of RAM, running a 64-bit operating system. During all tests, the number of iterations (generations) in the evolutionary process has been set to 100000. The frequency of applying crossovers vs. mutations has been set to 95% vs. 5%. It means that in each iteration exactly one crossover or mutation has been executed, while mutation has been activated only occasionally with probability 5%. The initial population has been assembled from three types of solutions of conventional MWIS problem instances. More precisely, for each particular scenario the corresponding (conventional) optimal solution has been included, as well as the greedy solution obtained by using the vertex-selecting rule (3.1) and the greedy solution obtained by using the rule based on plain weights. If such initial population turned out to be smaller than 30, additional chromosomes have been created from the original ones by applying mutations.

As we have just seen, our EA relies heavily on optimal solutions of conventional problem instances for particular scenarios. Indeed, such solutions are inserted into the initial population. But even more importantly, they are needed as parameters within the robust deviation or relative robust deviation objective function. For our relatively small problem instances, the required optimal conventional solutions could be computed exactly by CPLEX. However, for larger instances exact solving would become impossible due to NP-hardness of the conventional MWIS problem. Therefore, we have decided to perform the required computations only approximately, i.e. by a method that is applicable even for very large problem instances. It means that the so-called exact optimal solutions within the initial population are in fact high-quality approximations of those solutions. Also, the parameters within the robust objective functions used by the EA are in fact approximate.

To find high-quality approximate solutions of the involved conventional problem instances, we have used the same EA as for robust instances. Indeed, a conventional instance can be considered as a robust instance with only one scenario conforming to the absolute criterion of robustness. The initial population in such case is rather small, but as mentioned earlier, it can be increased by mutations. Since our EA partly relies on random choices, repeated executions of the same computational task usually do not produce the same results. Consequently, for any given conventional problem instance, the corresponding computation has been repeated 10 times and the best obtained solution has been retained.

As our EA is an approximate algorithm, the most important indicator of its performance is its accuracy. In our tests involving robust problem instances, we have measured accuracy by computing the relative errors of approximate solutions versus exact (truly optimal) solutions. More precisely, in each test we have computed relative difference between the robust objective-function value obtained by the EA solution and the corresponding optimal robust objective-function value assessed by CPLEX. Thereby the authentic version of the robust objective function has been used, whose parameters have been determined exactly by CPLEX.

In our tests, we have solved each of the 30 problem instances from each of the 2 test groups by each of the 20 crossover/mutation combinations, according to each of the 3 robustness criteria. As already mentioned, repeated executions of the same task usually do not produce the same solution due to randomized computing. Consequently, for any given problem instance, crossover/mutation combination and robustness criterion, the corresponding computation has been repeated 10 times.

The results of our tests are summarized in Tables 3.1 and 3.2. In fact, both tables present the same results, but in a slightly different way. In any table, six parts correspond to 3 robustness criteria vs. 2 test groups. Each part contains average errors for different combinations of crossovers and mutations.

The difference between Table 3.1 and 3.2 is as follows. In Table 3.1, the errors obtained through 10 repeated executions of the same computational task have been averaged. In Table 3.2, only the best (smallest) error obtained in 10 repeated executions has been recorded. In both tables, the collected values (average or best errors) have further been averaged over test groups. In this way, fairly reliable indicators of performance have been obtained. The results from Table 3.2 always look better than those from Table 3.1. Indeed, in many cases our EA finds near optimal solution in some tries, but then spoils the statistics in the remaining tries.

Table 3.1: Approximation accuracy of our EA - average relative errors

300 vertices, 30000 edges, 10 scenarios - absolute robustnes

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 5.61 % | 7.16 % | 6.98 % | 7.23 % | 6.41 % |
| MAVX | 5.06 % | 6.14 % | 6.66 % | 6.75 % | 5.74 % |
| RVX | 2.04 % | 2.19 % | 2.01 % | 2.18 % | 1.64 % |
| MRVX | 2.36 % | 2.19 % | 2.51 % | 2.19 % | 1.90 % |

300 vertices, 20000 edges, 10 scenarios - absolute robustness

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 7.79 % | 8.58 % | 8.98 % | 9.93 % | 9.81 % |
| MAVX | 7.5 % | 8.89 % | 8.97 % | 9.02 % | 8.66 % |
| RVX | 4.35 % | 4.39 % | 4.67 % | 4.76 % | 3.67 % |
| MRVX | 3.83 % | 3.53 % | 3.82 % | 3.78 % | 3.50 % |

300 vertices, 30000 edges, 10 scenarios - robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 10.63 % | 12.53 % | 12.96 % | 12.84 % | 12.06 % |
| MAVX | 10.17 % | 12.36 % | 12.28 % | 12.49 % | 11.45 % |
| RVX | 4.17 % | 4.69 % | 4.98 % | 5.10 % | 3.60 % |
| MRVX | 5.37 % | 5.55 % | 5.67 % | 5.50 % | 4.17 % |

300 vertices, 20000 edges, 10 scenarios - robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 14.07 % | 15.95 % | 16.66 % | 17.10 % | 16.16 % |
| MAVX | 13.93 % | 14.90 % | 15.02 % | 15.79 % | 14.32 % |
| RVX | 9.60 % | 9.93 % | 9.76 % | 8.82 % | 8.50 % |
| MRVX | 8.51 % | 8.53 % | 9.31 % | 8.77 % | 7.88 % |

| 300 vertices, 30000 edges, 10 scenarios - relative robust deviation | | | | | |
|---|---|---|---|---|---|
| | SRM | WIRM | WDRM | LSRM | CM |
| AVX | 9.05 % | 11.03 % | 11.73 % | 11.44 % | 10.71 % |
| MAVX | 9.54 % | 10.69 % | 11.09 % | 10.24 % | 9.30 % |
| RVX | 0.34 % | 0.25 % | 0.36 % | 0.37 % | 0.39 % |
| MRVX | 4.80 % | 4.80 % | 4.82 % | 5.11 % | 4.07 % |

| 300 vertices, 20000 edges, 10 scenarios - relative robust deviation | | | | | |
|---|---|---|---|---|---|
| | SRM | WIRM | WDRM | LSRM | CM |
| AVX | 13.66 % | 15.75 % | 15.77 % | 17.27 % | 15.98 % |
| MAVX | 13.48 % | 15.29 % | 15.12 % | 15.18 % | 14.83 % |
| RVX | 5.96 % | 6.49 % | 6.14 % | 7.87 % | 6.21 % |
| MRVX | 7.81 % | 7.79 % | 8.35 % | 8.33 % | 6.88 % |

Table 3.2: Approximation accuracy of our EA - best relative errors

| 300 vertices, 30000 edges, 10 scenarios - Absolute robustnes | | | | | |
|---|---|---|---|---|---|
| | SRM | WIRM | WDRM | LSRM | CM |
| AVX | 2.54 % | 4.47 % | 4.18 % | 4.83 % | 4.09 % |
| MAVX | 1.66 % | 3.58 % | 3.64 % | 3.85 % | 3.37 % |
| RVX | 0.08 % | 0.39 % | 0.15 % | 0.12 % | 0.09 % |
| MRVX | 0.78 % | 0.54 % | 0.67 % | 0.89 % | 0.54 % |

| 300 vertices, 20000 edges, 10 scenarios - Absolute robustness | | | | | |
|---|---|---|---|---|---|
| | SRM | WIRM | WDRM | LSRM | CM |
| AVX | 3.77 % | 4.49 % | 4.77 % | 5.87 % | 5.54 % |
| MAVX | 3.51 % | 4.40 % | 4.67 % | 4.11 % | 4.21 % |
| RVX | 1.16 % | 1.01 % | 1.37 % | 1.58 % | 0.97 % |
| MRVX | 1.07 % | 1.24 % | 0.83 % | 1.09 % | 1.24 % |

300 vertices, 30000 edges, 10 scenarios - Robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 4.64 % | 7.68 % | 9.60 % | 7.20 % | 6.61 % |
| MAVX | 4.64 % | 7.76 % | 7.92 % | 6.39 % | 4.96 % |
| RVX | 0.67 % | 0.48 % | 0.72 % | 1.14 % | 0.43 % |
| MRVX | 0.94 % | 0.64 % | 0.88 % | 0.65 % | 0.48 % |

300 vertices, 20000 edges, 10 scenarios - Robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| | SRM | WIRM | WDRM | LSRM | CM |
| AVX | 7.17 % | 8.82 % | 8.32 % | 12.20 % | 10.62 % |
| MAVX | 6.51 % | 7.77 % | 8.31 % | 8.25 % | 7.46 % |
| RVX | 3.36 % | 2.90 % | 3.45 % | 2.03 % | 2.78 % |
| MRVX | 2.78 % | 2.03 % | 3.45 % | 2.90 % | 3.26 % |

300 vertices, 30000 edges, 10 scenarios - Relative robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 3.78 % | 6.78 % | 6.97 % | 6.60 % | 6.67 % |
| MAVX | 4.71 % | 5.51 % | 7.35 % | 6.51 % | 4.31 % |
| RVX | 0.13 % | 0.18 % | 0.18 % | 0.13 % | 0.13 % |
| MRVX | 0.17 % | 1.72 % | 0.58 % | 0.87 % | 0.85 % |

300 vertices, 20000 edges, 10 scenarios - Relative robust deviation

| | SRM | WIRM | WDRM | LSRM | CM |
|---|---|---|---|---|---|
| AVX | 6.49 % | 8.59 % | 9.44 % | 10.26 % | 9.49 % |
| MAVX | 6.94 % | 8.70 % | 7.75 % | 6.98 % | 7.69 % |
| RVX | 1.02 % | 0.72 % | 0.84 % | 0.58 % | 0.84 % |
| MRVX | 2.97 % | 2.64 % | 2.86 % | 2.85 % | 1.62 % |

Let us now analyze in more detail the results obtained with the absolute robustness criterion. The analysis is based on the more rigorous Table 3.1. CM can be regarded as the best mutation. For both test groups the CM operator gives the lowest errors combined with the RVX and

MRVX crossovers and the lowest in general. The other mutation operators perform similarly but worse than CM. If we compare crossover operators, RVX and MRVX are better then the AVX and MAVX for both test groups. Indeed, for the first test group and in combination with the CM, both AVX and MAVX produce errors that are almost triple as big as those produced by RVX or MRVX. Putting it all together, the best average error in general for the first test group is 1.64%, and it is accomplished by RVX combined with CM. On the other hand, the best average error for the second test group is 3.50%, which is attained by MRVX combined with CM.

Now we analyze the results obtained with the robust deviation criterion, as presented by Table 3.1. Again, CM can be regarded as the best mutation. For both test groups the CM operator gives the lowest errors combined with the RVX and MRVX crossovers, and the lowest in general. The second best mutation seems to be the SRM - it produces the lowest errors when combined with the AVX and MAVX, and the second lowest errors when combined with other two crossover operators (except with RVX for the second test group). Further, if we compare crossover operators, we see that again RVX and MRVX give much lower errors than the AVX and MAVX, respectively. In general, the best average error for the first test group is 3.60%, and it is again accomplished by RVX combined with CM. Regarding the second test group, the best average error is 7.88%, which is again achieved by MRVX combined with CM. The average errors are noticeably greater for this robust criterion than for the absolute criterion. Indeed, the best average errors are around 2 times greater.

Finally we analyze the results obtained with the relative deviation criterion for robustness. Again, Table 3.1 is taken into account. For the first test group, it is very noticeable how the RVX crossover produces extremely low errors compared to the other 3 crossovers. Also, different choices of mutation do not have a large impact when combined with RVX. For the second test group, RVX is again the best crossover, but it is not so significantly better as for the first test group. When considering all combinations of operators, the lowest possible average errors are produced by the following combinations - RVX combined with WIRM for the first test group and RVX with SRM for the second. The concrete values are 0.25% and 5.96%, respectively. For the first time, CM does not give the lowest relative errors in general.

It is interesting that, for 4 out of 6 cases, the best mutation turns out to be CM. Remember that CM is an extreme variant of mutation, which creates a new chromosome by using vertices not used by the original chromosome, thus bringing more diversity to the population. For the other two cases, CM is not the best, but its error is very close to the lowest.

Note that for all three criteria of robustness and for both test groups our algorithm can assure average relative errors less than 8%. This is quite satisfactory if we take into account that such approximate solutions are obtained much faster than exact solutions. The average execution time of CPLEX and of our EA are summarized in Table 3.3 and 3.4, respectively. As already stated, CPLEX needs more time for sparser graphs because they allow more independent sets. In such case, our algorithm will produce a solution in around 2.8 seconds, which is around 690 times faster than CPLEX would do for robust deviation. The average execution time of our EA depends not only on graph density, but also on the chosen crossover and mutation operators. The values shown in Table 3.3 correspond to those combinations of operators that assure best accuracy, i.e RVX with CM or WIRM for the first test group and RVX or MRVX with CM or SRM for the second test group.

Table 3.3: Average CPU time in seconds needed by CPLEX to find exact solutions

| | Absolute robustness | Robust deviation | Relative robust deviation |
|---|---|---|---|
| 30000 edges | 197 | 69 | 75 |
| 20000 edges | 978 | 2064 | 1942 |

Table 3.4: Average CPU time in seconds needed by our EA

| | Absolute robustness | Robust deviation | Relative robust deviation |
|---|---|---|---|
| 30000 edges | 2.717 | 2.780 | 2.358 |
| 20000 edges | 3.136 | 3.230 | 2.814 |

To conclude, our tests indicate that performance and relative ranking of particular evolutionary operators are both influenced by robustness criteria and graph density. Indeed, for the first test group the best solutions are obtained by combining RVX with CM for asbsolute robustness and robust deviation, but RVX with WIRM for relative robust deviation. Further, for the second group the best solutions are produced by MRVX combined with CM for absolute robustness and robust deviation, but RVX with SRM for relative robust deviation. With most efficient combinations of evolutionary operators, the expected errors range between 0.25% and 8%. Finally, our algorithm turns out to be between 25 and 690 faster than the exact algorithm provided by CPLEX. The actual speedup again depends on the chosen robustness criterion and on the involved graph density.

# 4. Robust MWIS problem variants on trees

Although the conventional MWIS problem is NP-hard in general [12], it still can be solved in polynomial time on some special classes of graphs, such as trees or interval graphs or apple-free graphs. Indeed, the paper [8] specifies an algorithm for trees, with linear complexity in terms of number of vertices, which is based on dynamic programming. Next, in [11, 21, 25, 27] we can find polynomial algorithms for interval or apple-free graphs.

In this chapter we are concerned with solving robust variants of the MWIS problem on trees. So we are dealing with one of the special classes of graphs that are regarded as interesting for the reasons explained above. Since the conventional MWIS problem is NP-hard in general, it is clear that all its robust variants must also be NP-hard in general. But let us point out again that the conventional MWIS problem on the mentioned special classes can be solved in polynomial time. This gives rise to a hope that robust variants of the same problem on the same classes can also be solved more efficiently than in the general case, maybe even in polynomial time. To see if such hope is justified, it is necessary to analyze computational complexity of robust MWIS problem variants on particular types of graphs. Such analysis has already been done in [16, 30] for interval graphs - it turned out that almost all robust MWIS variants on interval graphs are NP-hard. To the best of our knowledge, there are no similar works in literature dealing with trees or apple-free graphs.

In this chapter we will first establish a relationship between trees and interval graphs. It will be shown that none of those two classes of graphs is a subset of the other. This is important because it means that the available complexity results from [16, 30] regarding interval graphs cannot be applied to trees. Thus a separate study of complexity is needed. In the second section of the chapter we will present our original complexity analysis dealing with robust MWIS problem variants on trees. Our results are analogous to those for interval graphs from [16, 30]. Consequently, they show that the considered problem variants on trees are again NP-hard. Thus again, we cannot expect exact polynomial-time solutions. In the remaining sections of this chapter we will therefore concentrate on a custom-designed *approximate* algorithm, which solves robust MWIS problem variants on trees more efficiently than the general evolutionary algorithm from Chapter 3.

## 4.1. Relationship between trees and interval graphs

In this section we demonstrate that none of the two considered classes of graphs is a subset of the other. Indeed, here are the definitions of both classes.

- An undirected graph $G$ is a *tree* if it is connected and acyclic. Or equivalently, an undirected graph $G$ with $n$ vertices is a *tree* if it is connected and its number of edges is $n-1$.

  A tree is usually organized in a hierarchy, so that one vertex is chosen to be the root, its neighbors become its children, remaining neighbors of children become children's children, etc. Then any vertex can have 0, 1 or more children. The root has no parent, and any other vertex has exactly one parent. Vertices with no children are called leaves.

- Let $I_i = [a_i, b_i]$, denote a closed interval, where $a_i, b_i \in \mathbb{R}$ and $a_i < b_i$. An undirected graph $G = (V, E)$ with $|V| = n$ vertices is called *interval graph* for a finite family $\mathscr{I} = \{I_1, I_2, ..., I_n\}$ of intervals on the real line if there is a one-to-one correspondence between $\mathscr{I}$ and $V$ such that two intervals in $\mathscr{I}$ have non-empty intersection if and only if their corresponding vertices in $V$ are adjacent to each other.

The fact that trees and interval graphs are two different classes is demonstrated by Figure 4.1, where we can see an interval graph that is not a tree, and also a tree that is not an interval graph.

Let us explain Figure 4.1 in more detail. The graph on the left is obviously not a tree because it has a cycle, but on the other hand it is a interval graph since it corresponds e.g. to the family $\mathscr{I} = \{[1,4], [2,3], [2,4]\}$. The graph on the right is obviously a tree, but it is not an interval graph since there exists no corresponding family of intervals. Namely, any attempt to construct

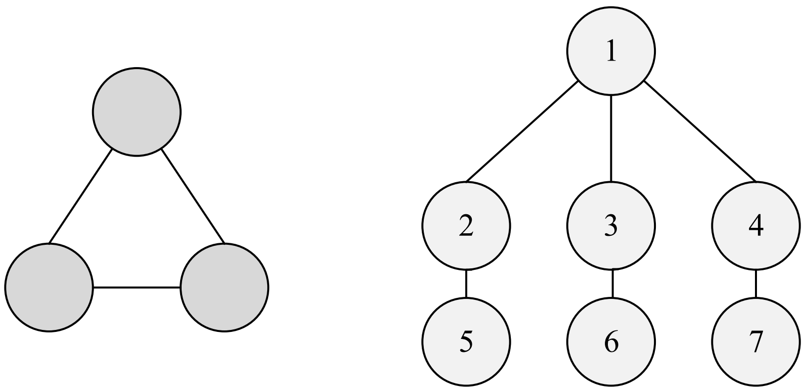

Figure 4.1: An interval graph that is not a tree, a tree that is not an interval graph

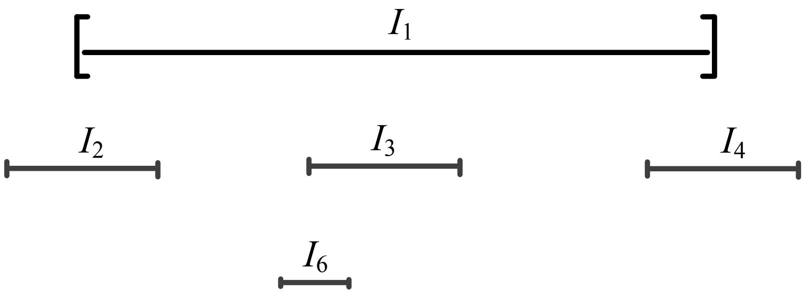

Figure 4.2: Demonstrating by contradiction that the tree from Figure 4.1 cannot be an interval graph

such a family leads to a contradiction. Indeed, let us try to construct such intervals. Denote with $I_i$ the interval corresponding to vertex $i$. Our attempt is illustrated by Figure 4.2.

- Consider first $I_2$, $I_3$ and $I_4$ - since their vertices are not adjacent, they should be disjunct and placed on the real line in some sequence. Without any loss of generality we can assume that $I_2$ is on the left, $I_3$ in the middle and $I_4$ on the right, as shown in the middle part of Figure 4.2 (otherwise we could renumber the vertices).

- Next consider $I_1$. Since vertex 1 is adjacent to vertices 2, 3 and 4, $I_1$ should overlap with all three intervals $I_2$, $I_3$ and $I_4$. More precisely, the left endpoint of $I_1$ cannot be larger than the right endpoint of $I_2$ since otherwise the two intervals would not overlap. Similarly, the right endpoint of $I_1$ cannot be smaller than the left endpoint of $I_4$. So the situation looks as shown in the upper part of Figure 4.2. A consequence is that $I_3$ must be a subset of $I_1$.

- Finally, we consider $I_6$. Since vertices 3 and 6 are adjacent, $I_6$ should overlap with $I_3$, as shown in the lower part of Figure 4.2. But since $I_3$ is a subset of $I_1$, it means that $I_6$ must also overlap with $I_1$. This is a contradiction with the fact that vertices 1 and 6 are not adjacent.

## 4.2. Complexity of robust MWIS problem variants on trees

A natural question one would like to answer is whether there exists an exact algorithm, similar to the one from [8], which would solve robust variants of the MWIS problem on trees in polynomial time. Unfortunately, the answer is in most cases negative (unless P=NP) due to the following theorems. They are proved by polynomial reduction of the standard *2-partition problem* to our robust variants. We start with the definition of 2-partition:

**Instance:** a list of positive integers $a_1, a_2, \ldots, a_n$.

**Question:** is there a subset of indices $S \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$ ?

It is well known [12] that the 2-partition problem is NP-complete.

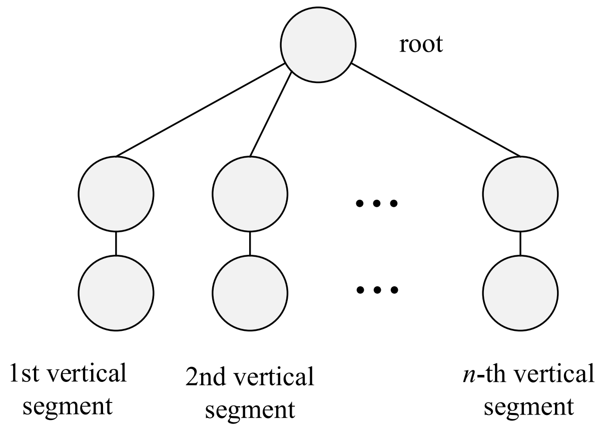

Figure 4.3: A tree that reduces a 2-partition problem instance to a MWIS problem instance

**Theorem 4.1.** Finding an absolute robust solution of the MWIS problem on trees is NP-hard, even with only two discrete scenarios.

*Proof.* Let us consider an instance of the 2-partition problem specified by positive integers $a_1, a_2, \ldots, a_n$. Denote with $T$ the sum $\sum_{i=1}^{n} a_i$. We construct the corresponding MWIS problem instance with 2 scenarios on the tree shown in Figure 4.3. The first scenario is specified as follows:

- The root has weight 0.
- In the $i$-th vertical segment the upper vertex has weight $a_i$ and the lower vertex has weight 0.

The second scenario is specified as follows:

- The root has weight 0.
- In the $i$-th vertical segment the lower vertex has weight $a_i$ and the upper vertex has weight 0.

A nontrivial independent set (i.e. one that cannot be extended by adding more vertices) is formed so that exactly one (upper or lower) vertex is chosen in each vertical segment from Figure 4.3. The root may or may not be chosen, but it is irrelevant since its weight is 0. Here follow some observations.

- If the selected independent set contains many upper vertices, it will have a large weight under the first scenario, but a small weight under the second scenario. Thus its minimum weight over scenarios will be small, so that it will not be an absolute robust solution.
- If the selected independent set contains many lower vertices, it will have a large weight under the second scenario, but a small weight under the first scenario. Again, its minimum weight over scenarios will be small, so that it again will not produce an absolute robust solution.
- An absolute robust solution (max-min) is achieved when the sum of weights of the chosen upper vertices is approximately equal to the sum of weights of the chosen lower vertices, since then the minimum over scenarios is as large as possible.

From the above observations we can deduce the following. If the robust objective function value (max-min) happens to be exactly $T/2$, then a 2-partition of $a_1, a_2, \ldots, a_n$ exists. If it is $< T/2$, then a 2-partition does not exist.

It means that by solving the robust MWIS problem instance we obtain the solution of the 2-partition instance. ■

**Theorem 4.2.** Finding a robust deviation solution of the MWIS problem on trees is NP-hard, even with only two discrete scenarios.

*Proof.* Let us consider an instance of the 2-partition problem specified by positive integers $a_1, a_2, \ldots, a_n$. Denote again with $T$ the sum $\sum_{i=1}^{n} a_i$. We construct the corresponding MWIS problem instance in the same way as in the proof of Theorem 4.1, i.e. the tree looks as shown in Figure 4.3 and the two scenarios are the same. Nontrivial independent sets are also formed in the same way as before. Now we can observe the following.

- The optimal conventional solution under the first scenario is obtained by choosing the upper vertex within each vertical segment from Figure 4.3. The weight of that solution is $T$.
- The optimal conventional solution under the second scenario is obtained by choosing the lower vertex within each vertical segment from Figure 4.3, and its weight is again $T$.
- Let us consider any independent set $X$. Let $S$ be the set of indices of vertical segments from Figure 4.3 where $X$ has chosen the upper vertex. Then the "regret" for $X$ under the first scenario is equal to $T - \sum_{i \in S} a_i$, and the regret for $X$ under the second scenario is $T - \sum_{i \notin S} a_i$.
- If $X$ contains many upper vertices, it will have a small regret under the first scenario, but a large regret under the second scenario, so that its maximal regret over both scenarios will be large.
- If $X$ contains many lower vertices, it will have a large regret under the first scenario and a small regret under the second scenario. Again, its maximal regret over both scenarios will be large.
- A robust deviation solution (min-max regret) is achieved when $\sum_{i \in S} a_i$ is approximately equal to $\sum_{i \notin S} a_i$, since then the maximal regret over both scenarios is as small as possible and $\approx T/2$.

From the above observations we can deduce the following. If the robust objective function value (min-max regret) happens to be exactly $T/2$, then a 2-partition of $a_1, a_2, \ldots a_n$ exists. If it is $> T/2$, then a 2-partition does not exist.

Thus by solving the constructed robust MWIS problem instance we solve the given 2-partition instance. ■

**Theorem 4.3.** Finding a relative robust deviation solution of the MWIS problem on trees is NP-hard, even with only two discrete scenarios.

*Proof.* It is almost the same as for Theorem 4.2. For a given 2-partition problem instance specified by positive integers $a_1, a_2, \ldots, a_n$ we construct the same MWIS problem instance again (the same tree and scenarios). Nontrivial independent sets are formed in the same way as before, and the optimal conventional solutions under particular scenarios are the same. $T$ again stands for $\sum_{i=1}^{n} a_i$. But now we can observe the following.

- Let us consider any independent set $X$, and let $S$ be the set of indices of vertical segments from Figure 4.3 where $X$ has chosen the upper vertex. Then the relative regret for $X$ under the first scenario is $(T - \sum_{i \in S} a_i)/T$, while the relative regret under the second scenario is $(T - \sum_{i \notin S} a_i)/T$. The division with $T$ is legal since $T$ is a sum of positive integers, thus it is greater than 0.

- A relative robust deviation solution (min-max relative regret) is achieved again when $\sum_{i \in S} a_i$ is approximately equal to $\sum_{i \notin S} a_i$, since then the maximal relative regret over both scenarios is as small as possible and $\approx (T - T/2)/T = 1/2$.

From the above observations we can deduce the following. If the robust objective function value (min-max relative regret) happens to be exactly $1/2$, then a 2-partition of $a_1, a_2, \ldots a_n$ exists. If it is $> 1/2$, then a 2-partition does not exist.

Thus by solving the constructed robust MWIS problem instance we solve the given 2-partition instance. ■

With the above three theorems, we have proved that the absolute robust variant and the (relative) robust deviation variants of the MWIS problem on trees are NP-hard. Since the involved three robustness criteria are special cases of the $OWA_A$, $OWA_D$ and $OWA_R$ criteria, respectively, we immediately obtain the following consequence.

**Corollary 4.4.** Finding an $OWA_A$ or $OWA_D$ or $OWA_R$ solution of the MWIS problem on trees is NP-hard, even with only two discrete scenarios.

Next, we can combine the complexity results from this section with the equivalency results from Sections 2.1 and 2.2. In this way we can immediately prove some assertions regarding robust variants of the MWVC problem on trees. For instance, the combination of Proposition 2.3 with Theorem 4.1 brings the following consequence.

**Corollary 4.5.** Finding an absolute robust solution of the MWVC problem on trees is NP-hard, even with only two discrete scenarios.

The corollary above follows because the MWIS problem instance used in the proof of Theorem 4.1 can (polynomially) be reduced to the corresponding MWVC problem instance. Such a reduction is correct since the involved instances satisfy the restriction from Proposition 2.3 regarding scenarios. Similarly, Proposition 2.8 can be combined with Theorem 4.2, thus giving an additional consequence.

**Corollary 4.6.** Finding a robust deviation solution of the MWVC problem on trees is NP-hard, even with only two discrete scenarios.

It is easy to see that an analogue claim regarding relative robust deviation solution of the MWVC problem on trees is also true. Such claim can be proved directly by a similar (but slightly modified) construction as in Theorem 4.3.

So far we have assumed that uncertainty in our robust MWIS problem variants is captured through discrete scenarios. Let us now say a few words about situations where uncertainty is expressed by intervals.

- It is clear that the combination of absolute robustness with interval uncertainty on trees gives a variant of the MWIS problem that can be solved in polynomial time - this fact has already been stated within Corollary 2.16.
- A more complicated case is when (relative) robust deviation is combined with interval uncertainty. At this moment we do not know if such variants of the MWIS problem on trees are polynomially solvable or not. We must leave this issue as an open problem for further research.

At the end of this section, let us note that there exists an alternative way of proving Theorems 4.1, 4.2 and 4.3. It is based on polynomial reduction of the *even-odd partition problem* (rather than 2-partition) to our robust variants. Here is the definition of even-odd partition.

**Instance:** a multiset of positive integers $a_i$, $1 \leq i \leq 2n$, such that $\sum_i a_i = T$.
**Question:** can the multiset of all $a_i$-s be divided into two disjoint parts $A_1$ and $A_2$ such that $\sum_{a_i \in A_k} a_i = T/2$ for $k = 1, 2$ and precisely one of $a_{2i-1}, a_{2i}$ belongs to $A_1$ for $1 \leq i \leq n$.

It is well known [12] that the even-odd partition problem is NP-complete.

Now we give the alternative proof for Theorem 4.1. Analogous modifications can also be done for Theorems 4.2 and 4.3.

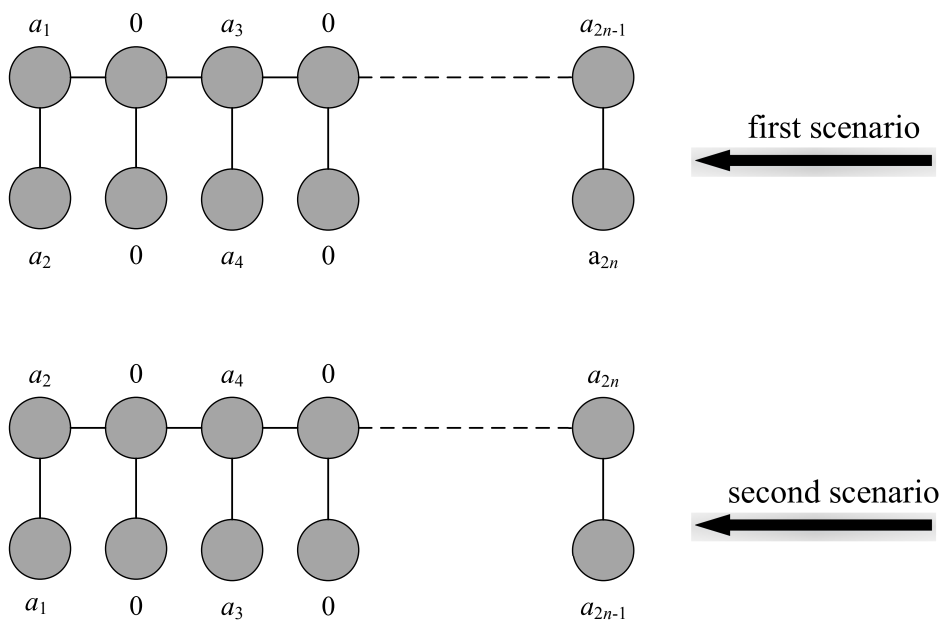

Figure 4.4: A tree that reduces an even-odd partition problem instance to a MWIS problem instance

*Proof.* Let us consider an instance of the even-odd partition problem specified by positive integers $a_1, a_2, \ldots, a_{2n}$. Denote the sum $\sum_{i=1}^{2n} a_i$ with $T$. We construct the corresponding MWIS problem instance with 2 scenarios on a tree with $4n-2$ vertices. The details are specified by Figure 4.4.

We want to find an independent set whose minimum weight under both scenarios is as large as possible. Inserting a vertex with weight 0 into a set obviously does not affect the weight of that set. We will consider only such independent sets that among two adjacent vertices with weight 0 (see Figure 4.4) always pick the lower vertex. Namely, if some upper vertex with weight 0 is chosen, then its left and right neighbor cannot be chosen. On the other hand, when a lower vertex with weight 0 is chosen, then any of the two vertices in a neighboring vertical segment can be chosen, which gives more options to achieve a larger total weight.

Let $X$ be a nontrivial independent set whose vertices with weight 0 are lower vertices. By the structure of the graph it is clear that for each $i$, $1 \leq i \leq n$, $X$ contains exactly one among vertices with weights $a_{2i-1}$, $a_{2i}$. Let $A_1$ be the collection of weights of vertices chosen by $X$ according to the first scenario. Let $A_2$ be the complement of $A_1$ (regarding the whole collection of $a_i$-s). Then the weight of $X$ under the first scenario is $\sum_{a_i \in A_1} a_i$, and the weight of $X$ under the second scenario is $\sum_{a_i \in A_2} a_i$. Here are some observations.

- If one of these two sums is large, the other one must be small, and the minimum of the two sums will be small.

- The minimum of the two sums will be as large as possible if the two sums are roughly equal.

From the above observations we can deduce the following. If the robust objective function value happens to be exactly $T/2$, then an even-odd partition exists. If it is $< T/2$, then an even-odd partition does not exist.

Consequently, by solving the constructed robust MWIS problem instance we obtain the solution of the given even-odd partition problem instance. ■

## 4.3. THE EXACT ALGORITHM FOR SOLVING THE CONVENTIONAL MWIS PROBLEM ON TREES

In this section we will briefly describe the exact algorithm for solving the conventional MWIS problem on trees from [8]. Namely, our new algorithm from Section 4.4 is inspired by some procedures from the conventional one. As previously mentioned, the algorithm from [8] has linear complexity, moreover it visits each vertex exactly once. In more detail, the algorithm visits each vertex $v_i$ in a given tree just once, and it constructs two corresponding independent sets:

- An independent set for the subtree rooted at $v_i$ that has the greatest weight, and that contains $v_i$. Such set will be called the *inclusive* independent set.
- An independent set for the subtree rooted at $v_i$ that has the greatest weight, and that does not contain $v_i$. Such set will be called the *exclusive* independent set.

In its $i$-th step the algorithm constructs the above described two independent sets for vertex $v_i$ by using previously constructed independent sets for children of $v_i$. th construction is done in the following way.

- The inclusive independent set for $v_i$ is obtained as the union of exclusive independent sets for children of $v_i$, plus $v_i$ itself.
- The exclusive independent set for $v_i$ is obtained as a union of either inclusive or exclusive independent sets for children of $v_i$. For each child, the one with greater weight is chosen.

As its final solution for the whole tree, the algorithm chooses a better one among the two independent sets obtained for the root. The comparison is again done according to weights. It is easy to prove that the described algorithm is correct, i.e. it really constructs a solution that must be optimal in the conventional sense.

In the remaining part of this section we will present a concrete example illustrating how the algorithm works.

**Example 4.7.** Figure 4.5 shows a weighted tree with 14 vertices. Each vertex is labeled with its weight. We construct the inclusive and exclusive independent set for each vertex, as presented in Table 4.1.

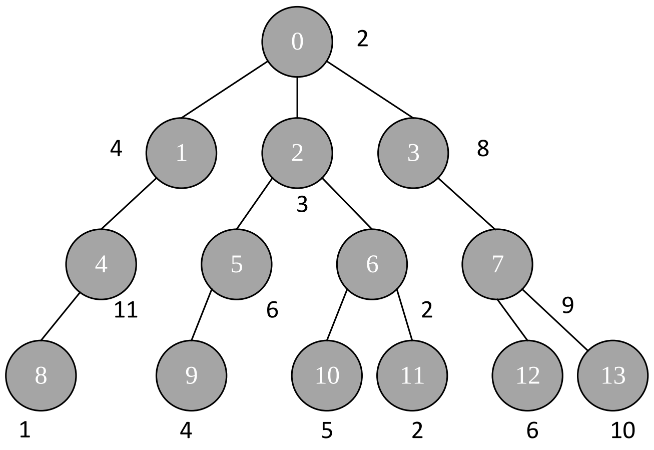

Figure 4.5: A weighted tree with 14 vertices

Now let us explain the data from Table 4.1 in more detail. Vertices 8 to 13 are leaves. Obviously, for each leaf its exclusive independent set is empty, while its inclusive independent set contains only that leaf. After inclusive/exclusive independent sets of leaves are determined, the algorithm continues with vertices on the next level, i.e. vertices 4 to 7. For each of those vertices its inclusive/exclusive independent set is determined by the rules described previously. After that, the algorithm proceeds with the next level i.e. vertices 1 to 3, $\ldots$, until finally it reaches the root. For example, to find the exclusive independent set for the root, the algorithm must choose between $\{1,8\}$ and $\{4\}$, $\{2,9,10,11\}$ and $\{5,10,11\}$, and finally between $\{3,12,13\}$ and $\{12,13\}$.

The algorithm ends with choosing either exclusive or inclusive independent set obtained for the root. In this way the final solution is produced. In our case, $\{0,4,5,10,11,12,13\}$ has weight 42, while $\{2,3,4,9,10,11,12,13\}$ has weight 49. Therefore, the algorithm outputs the set $\{2,3,4,9,10,11, 12,13\}$.

Table 4.1: The inclusive and exclusive independent set for vertices of the tree from Figure 4.5 obtained by the exact algorithm

| vertex | inclusive | exclusive |
|---|---|---|
| 13 | $\{13\}$ | $\emptyset$ |
| 12 | $\{12\}$ | $\emptyset$ |
| 11 | $\{11\}$ | $\emptyset$ |
| 10 | $\{10\}$ | $\emptyset$ |
| 9 | $\{9\}$ | $\emptyset$ |
| 8 | $\{8\}$ | $\emptyset$ |
| 7 | $\{7\}$ | $\{12,13\}$ |
| 6 | $\{6\}$ | $\{10,11\}$ |
| 5 | $\{5\}$ | $\{9\}$ |
| 4 | $\{4\}$ | $\{8\}$ |
| 3 | $\{3,12,13\}$ | $\{12,13\}$ |
| 2 | $\{2,9,10,11\}$ | $\{5,10,11\}$ |
| 1 | $\{1,8\}$ | $\{4\}$ |
| 0 | $\{0,4,5,10,11,12,13\}$ | $\{2,3,4,9,10,11,12,13\}$ |

## 4.4. The population algorithm for solving robust variants of the MWIS problem on trees

In the Section 4.2 it has been shown that most of the considered robust variants of the MWIS problem on trees are NP-hard. It means that exact algorithms for solving such variants would take too much time and space. So it makes more sense to focus on *approximate* algorithms. In this section we propose an approximate algorithm for solving robust MWIS problem variants on trees. It turns out to be fast and accurate. It takes into account the special structure of the involved graph (i.e. a tree).

The algorithm presented in this section will be called *population* algorithm. Similar to the exact algorithm from the previous section, it also visits vertices of a tree and constructs independent sets for the corresponding subtrees. But unlike the previous algorithm, it does not produce only one exclusive or inclusive set per vertex, but collections of such sets.

More precisely, for a given tree the population algorithm visits each vertex $v_i$ of that tree exactly once and constructs two corresponding collections of independent sets.

- A collection of independent sets for the subtree rooted at $v_i$ that are considered "good" according to the chosen robustness criterion. Thereby each of those independent sets contains $v_i$. Such collection is called the *inclusive population*.

- A collection of independent sets for the subtree rooted at $v_i$ that are considered "good" according to the chosen robustness criterion. Thereby none of those independent sets contains $v_i$. Such collection is called the *exclusive population*.

In its $i$-th step the algorithm constructs the above described two populations for vertex $v_i$ by using previously constructed populations for children of $v_i$. The construction is done in the following way.

- A member of the inclusive population for $v_i$ is assembled as a union, where for each child of $v_i$ one member of its exclusive population is added. Vertex $v_i$ is also put into the union. The selection of particular members from children's populations is done randomly, but not with equal probability. Independent sets that are better according to the used robustness criterion have more chance to be chosen. In more detail, all independent sets are represented by sub-segments where their lengths correspond to objective-function values of that sets. Next, we generate a uniformly distributed random number which falls into the whole segment. The random number will more likely fall into a sub-segment with greater length. An example of the mentioned sub-segments is given by Figure 4.6.

- A member of the exclusive population for $v_i$ is created as a union, where for each child of $v_i$ one member of its inclusive or exclusive population is added. Again, independent sets from children's populations are chosen randomly, but those sets that are better according to the used robustness criterion are more likely to be selected. Random choice is implemented in the same way as for members of inclusive population.

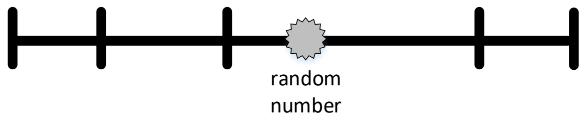

Figure 4.6: A segment divided in 4 sub-segments. A random value has fallen into the 3th sub-segment.

Notice that randomness does not ensure uniqueness of solutions in the inclusive or exclusive population. Indeed, some members could appear more than once if they are more likely to be fit. Comparing a newly created member with a list of already existing members would be an expensive operation. Hence, allowing to have copies of some solutions both requires less computing time and gives advantage to better solutions, i.e. it does not force filling a population with different solutions when they are bad just for the sake of the uniqueness.

As its final solution for the whole tree, our population algorithm selects the best independent set found within both populations that have been constructed for the root. The comparison of sets is again done according to the involved robustness criterion.

The described algorithm is called *population* algorithm since it creates populations of independent sets for each subtree of a tree. Its complexity is $\mathcal{O}(n)$, where $n$ is again the number of vertices in the three.

For calculating the objective function in case of (relative) robust regret we need optimal solutions for the corresponding conventional problems, i.e. we need optimal solution values for each scenario. Those values are obtained by using the exact algorithm from Section 4.3. Although each step of the algorithm finds a solution in a subtree, its value is evaluated as it was a solution for the whole tree. In more detail, for each scenario the regret of some solution in a subtree is calculated as difference between the global optimal value for that scenario and the solution value. Similary, relative regret is calculated as difference between the global optimal value and the solution value divided by the global optimal value. There are two reasons why we prefer global rather than local optima:

- It requires less memory. Moreover, it would be almost impossible to calculate optimal solutions in all subtrees in case of a large tree.
- We tend to find the global robustly optimal solution, hence comparing ”partial” solutions to global conventional solutions would hopefully lead more directly to the desired goal.

In order to assure good performance of the described algorithm, it is very important to choose the right size for populations. If populations are too small, there will be not enough diversity of solutions to be carried from one level of the tree to the next level. On the other hand, if populations are too big, the algorithm will spend too much memory and time.

For vertices at lower levels of a tree, such as leaves, it does not make sense to have large populations. Indeed, the exclusive population for a leaf can contain only the empty set, and the inclusive population can have just one set containing only that leaf. In our current implementation of the algorithm, the population size first grows with each level until it reaches 12, then it is fixed until the end of computation. More precisely, the exclusive population for a vertex $v_i$ will have the size equal to the sum of sizes of exclusive populations for children of $v_i$. When such a size exceeds 12, it is reduced to 12. The same rule also applies for inclusive populations.

Further, in order to assure an even better accuracy, our current implementation of the algorithm forces two greedy choices of independent sets within any inclusive population. More precisely:

- An inclusive population must have a member that is obtained as a union of best members from the respective children's exclusive populations. The comparison of members within any child population is done according to the involved robustness criterion.

- An inclusive population must have the so-called average-best member. It is the optimal solution of a conventional (nonrobust) problem instance where the weight of any vertex is set to the average of its weights over all scenarios. According to the exact algorithm from Section 4.3, the average-best member is easily obtained as a union of average-best members from the respective children's exclusive populations.

Similar two greedy choices are also done during the construction of an exclusive population. To summarize, in our present implementation any population with size 12 must contain the described 2 "greedy" independent sets, and only the remaining 10 members are created by random choice.

Finally, let us note that our population algorithm is interesting because it combines characteristics of three different algorithm paradigms:

1. *Dynamic programming*. We calculate a table of partial solutions. We use results of the previous calculations within subsequent calculations.

2. *Greedy approach*. We implicitly assume that an independent set will be a “good” solution for the whole tree if it is assembled from parts that are “good” solutions for subtrees of that tree.

3. *Evolutionary computing*. We use populations of solutions instead of single solutions. Our populations evolve during algorithm execution. Better solutions have more chance to survive and evolve.

In the remaining part of this section we will present a concrete example illustrating how our algorithm works.

**Example 4.8.** We will consider the tree from Figure 4.7. It is the similar to the tree from Figure 4.5, but it has one extra scenario. For the sake of simplicity, the population size will be reduced to 5. In more detail, the first member is a union of best members from the respective children’s populations. Further, the second member is a union of average-best members from the respective children’s populations and the remaining 3 members are created by random choice. We will use the absolute robustness. One possible outcome of the algorithm is represented in Table 4.2.

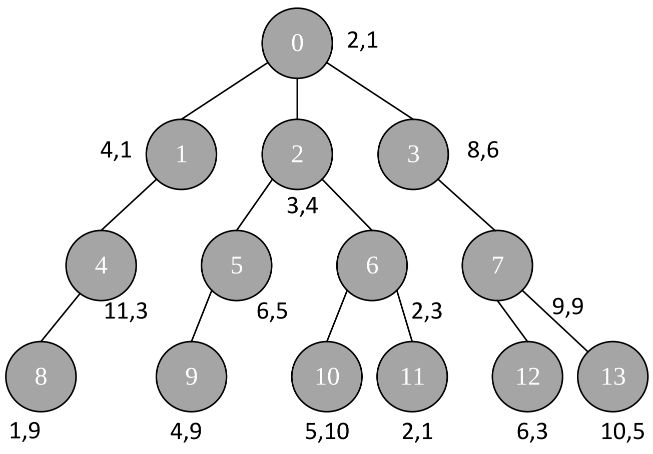

Figure 4.7: A tree with 14 vertices and 2 scenarios for vertex weights

Let us now analyze Table 4.2 in more detail. As stated before, the inclusive and exclusive population for a leaf are the set consisting of that leaf or an empty set, respectively. It is also easy to create both populations for vertices from 4 to 7. Vertices from 4 to 7 only have leaves for their children, which means that they have only one option how to construct a member of their inclusive population: they must put themselves into a single-element set. The tree from

Figure 4.5 is rather simple, but if we had a the tree with branches of different lengths, we would have more versatile populations.

Although for vertices in higher levels such as 0 and 2 we have more options for population members, some members are appearing more that once. It is because they are more fit for global solution and again because the tree from Figure 4.5 is small. In a bigger tree we would again have more versatile populations.

The algorithm ends with choosing either the inclusive or exclusive member from the populations obtained for the root. Among all possible independent sets, $\{1,2,3,8,9,10,11,12,13\}$ has the greatest robust (max-min) objective function value. In the considered case, the algorithm has found a solution which is trully optimal, i.e. the found solution coincides with the exact solution. It gives hope that the algorithm will also perform well on bigger and more complex examples. The actual test results will be given in Section 4.5.

Table 4.2: The inclusive and exclusive populations for vertices of the tree from Figure 4.7 obtained by our population algorithm according to the absolute robustness criterion

| vertex | inclusive population members | exclusive population members |
|---|---|---|
| 13 | $\{13\}$ | $\emptyset$ |
| 12 | $\{12\}$ | $\emptyset$ |
| 11 | $\{11\}$ | $\emptyset$ |
| 10 | $\{10\}$ | $\emptyset$ |
| 9 | $\{9\}$ | $\emptyset$ |
| 8 | $\{8\}$ | $\emptyset$ |
| 7 | $\{7\},\{7\},\{7\},\{7\},\{7\}$ | $\{12,13\},\{12,13\},\{12,13\},\{12,13\},\{13\}$ |
| 6 | $\{6\},\{6\},\{6\},\{6\},\{6\}$ | $\{10,11\},\{10,11\},\{10,11\},\{10\},\{10,11\}$ |
| 5 | $\{5\},\{5\},\{5\},\{5\},\{5\}$ | $\{9\},\{9\},\{9\},\{9\},\{9\}$ |
| 4 | $\{4\},\{4\},\{4\},\{4\},\{4\}$ | $\{8\},\{8\},\{8\},\{8\},\{8\}$ |
| 3 | $\{3,12,13\},\{3,12,13\},\{3,12,13\},$ $\{3,12,13\},\{3,12,13\}$ | $\{7\},\{12,13\},\{7\},\{7\},\{12,13\}$ |
| 2 | $\{2,9,10,11\},\{2,9,10,11\},\{2,9,10,11\},$ $\{2,9,10,11\},\{2,9,10,11\}$ | $\{5,10,11\},\{9,10,11\},\{9,10,11\},$ $\{9,10,11\},\{6,9\}$ |
| 1 | $\{1,8\},\{1,8\},\{1,8\},\{1,8\},\{1,8\}$ | $\{4\},\{4\},\{8\},\{8\},\{4\}$ |
| 0 | $\{0,4,5,7,10,11\},\{0,4,9,10,11,12,13\},$ $\{0,4,7,9,10,11\},\{0,4,7,9,10,11\},$ $\{0,4,5,7,10,11\}$ | $\{1,2,3,8,9,10,11,12,13\},$ $\{1,2,3,8,9,10,11,12,13\},$ $\{3,4,9,10,11,12,13\},$ $\{2,3,4,9,10,11,12,13\},$ $\{1,2,3,8,9,10,11,12,13\}$ |

## 4.5. Testing and results

As already mentioned earlier, to the best of our knowledge there are no benchmarks for robust variants of the MWIS problem, specially not on trees. Therefore we have generated our own 9 test groups, each comprising 30 problem instances. Those instances are based on random trees consisting of 30000, 60000 and 90000 vertices. As we wanted to test different tree configurations, for each number of vertices we have produced 3 groups of trees. Thereby vertices from the first, second and third such group can have maximum 5, 10 and 15 children, respectively. The chosen problem instances can be regarded as large enough to be nontrivial, but still small enough to be solved exactly by CPLEX.

Let us note that the involved problem instances are much larger in terms of number of vertices than those from Chapter 3 used for testing the evolutionary algorithm. Indeed, in Chapter 3 we have used random graphs with 300 vertices and 30000 or 20000 edges, while now we have at least 100 times more vertices and up to 4.5 times more edges. Such an increase in size is possible since CPLEX makes adequate preprocessing, thus reducing its execution time when working with trees. Similarly, the population algorithm from this chapter is customized for trees and therefore much faster than the general evolutionary algorithm from Chapter 3. For such reasons, the algorithm from this chapter and the evolutionary algorithm from Chapter 3 are hard to compare, they are simply designed for different purposes. In our tests we have compared the algorithm from this chapter only with CPLEX.

In all 9 test groups each problem instance comprises 10 discrete scenarios for vertex weights. thereby each individual weight ranges between 1 and 1000. Again as in Chapter 3, we did not want to have vertices with extremely large weights since they would make optimal solutions too predictable. A full specification of all problem instances can be found in Appendix B of this dissertation.

The algorithm from this chapter has been implemented in the Java programming language [24]. The program is available in Appendix A of this thesis. This time we have not used C++ as in Chapter 3 since we wanted to take advantage of the garbage collector implemented within Java. Garbage collection is needed because our population algorithm uses a lot of memory to store populations of independent sets. When the algorithm moves to a higher level of a tree, the memory used for the previous level must be deallocated, otherwise we would run out of memory. Further, the algorithm from this chapter has been tested on a computer with an Intel Core i5-6600K @ 3.50 GHz processor and 16 GB of RAM, running a 64-bit operating system. It is the same computer that has already been used in Chapter 3. Also, the same machine has been used in both chapters for running CPLEX.

Some robust variants of the MWIS problem (i.e. deviation and robust deviation variant) require solutions of the conventional MWIS problem instances that correspond to particular scenarios. As we mentioned earlier, such conventional solutions can easily be calculated by using the exact polynomial-time algorithm from Section 4.3. Indeed, the conventional MWIS problem is polynomially solvable on trees and there is no need for approximations.

As our algorithm from this chapter is an approximate algorithm, the most important indicator of its performance is its accuracy. Again, in our tests involving robust problem instances, we have measured the accuracy by computing the relative errors of approximate solutions versus exact (truly optimal) solutions. More precisely, in each test we have computed relative difference between the robust objective-function value achieved with our approximate solution and the corresponding optimal robust objective-function value assessed by CPLEX. Thereby the correct version of the robust objective function has been used, whose parameters have been determined exactly by the polynomial-time algorithm from Section 4.3.

In our tests, we have solved each of the 30 problem instances from each of the 9 test groups according to each of the 3 robustness criteria: absolute robustness and (relative) robust deviation. As the population algorithm is nondeterministic, its repeated execution on the same input data usually does not produce the same solutions. Therefore all computations with the population algorithm have been repeated 10 times.

The results of our tests regarding accuracy are summarized in Tables 4.3 and 4.4. Both tables present the same results, but in a slightly different way. In any table, 3 parts correspond to 3 robustness criteria. Each part contains average errors for 9 different test groups, i.e. 9 different combinations of the number of vertices versus the maximum number of children per vertex.

The difference between Table 4.3 and 4.4 is as follows. In Table 4.3, the errors obtained through 10 repeated executions of the same computational task have been averaged. In Table 4.4, only the best (smallest) error obtained in 10 repeated executions has been recorded. In both tables, the collected values (average or best errors) have further been averaged over test groups.

Table 4.3: Accuracy of the population algorithm - average relative errors

| population algorithm - absolute robustness | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 0.25 % | 0.16 % | 0.12 % |
| 60000 vertices | 0.21 % | 0.13 % | 0.08 % |
| 90000 vertices | 0.17 % | 0.11 % | 0.08 % |

| population algorithm - robust deviation | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 5.71 % | 6.37 % | 8.05 % |
| 60000 vertices | 4.18 % | 5.78 % | 6.72 % |
| 90000 vertices | 4.05 % | 5.55 % | 5.37 % |

| population algorithm - relative robust deviation | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 5.73 % | 6.35 % | 7.85 % |
| 60000 vertices | 4.19 % | 5.78 % | 6.53 % |
| 90000 vertices | 4.07 % | 5.54 % | 5.32 % |

Let us analyze Table 4.3 in more detail. First we concentrate on absolute robustness. With that criterion the algorithm performs well. Indeed, average relative errors are less than 0.3%. We see that errors become lower when the number of vertices increases. It means that our algorithm is able to solve very large problem instances accurately. Also, it is interesting to note that errors drop when the maximum number of children increases. More children means fewer levels in trees. Although errors are smaller for outspread trees, we pay for this with longer execution times, as it will be seen in Table 4.5.

Next we analyze the results from Table 4.3 regarding robust deviation. The algorithm still works well since it produces errors between 4% and 8%. Moreover, errors again become lower when the number of vertices increases - this is similar as in the case of absolute robustness. However, contrary to absolute robustness, errors now increase if the number of children becomes larger.

Finally, we analyze the results from Table 4.2 regarding relative robust deviation. We see

Table 4.4: Accuracy of the population algorithm - best relative errors

| population algorithm - absolute robustness | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 0.24 % | 0.16 % | 0.12 % |
| 60000 vertices | 0.21 % | 0.13 % | 0.08 % |
| 90000 vertices | 0.17 % | 0.11 % | 0.08 % |
| **population algorithm - robust deviation** | | | |
| | 5 children | 10 children | 15 children |
| 30000 vertices | 5.7 % | 6.37 % | 8.05 % |
| 60000 vertices | 4.16 % | 5.78 % | 6.70 % |
| 90000 vertices | 4.04 % | 5.53 % | 5.37 % |
| **population algorithm - relative robust deviation** | | | |
| | 5 children | 10 children | 15 children |
| 30000 vertices | 5.59 % | 6.20 % | 7.59 % |
| 60000 vertices | 4.17 % | 5.67 % | 6.34 % |
| 90000 vertices | 4.00 % | 5.46 % | 5.20 % |

that those results are similar to the ones with robust deviation.

If we compare the best relative errors for the population algorithm from Table 4.3 with the corresponding average relative errors from Table 4.2, we see that there is no significant difference. Although the population algorithm is randomized, its solutions do not show much diversity. It seems that the algorithm is quite firmly guided by its greedy components. There is nothing similar to mutation which would bring more volatility.

The obtained results become even more satisfactory when we take the execution time into account. Tables 4.5, 4.6 and 4.7 present the execution times for all three robustness criteria. Each table has two parts, which correspond to the population algorithm and CPLEX, respectively.

From Tables 4.4, 4.5 and 4.6 we can see that the population algorithm is considerably faster than CPLEX. The speedup is between 70 and 250 for absolute robustness, between 60 and 850 for robust deviation, and from 45 to 425 for relative robust deviation.

Table 4.5: Average CPU time is seconds - population algorithm versus CPLEX - absolute robustness

| population algorithm - absolute robustness | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 0.176 | 0.154 | 0.134 |
| 60000 vertices | 0.357 | 0.292 | 0.268 |
| 90000 vertices | 0.552 | 0.452 | 0.408 |

| CPLEX - absolute robustness | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 17.417 | 32.336 | 33.582 |
| 60000 vertices | 25.671 | 28.138 | 37.576 |
| 90000 vertices | 37.576 | 34.802 | 41.975 |

We can also notice that CPLEX needs more time for (relative) robust deviation than for absolute robustness. Also, CPLEX spends much more time when the number of children is small. This is according to our expectations because in such situations exact branch-and-bound methods used by CPLEX become more demanding.

At the end of this section let us mention that we have also tested our algorithm on trees with only a few hundreds of vertices. On such small trees, it often finds exact solutions, i.e. the same solutions as CPLEX. But then CPLEX is also extremely fast, so that in such cases our approximate algorithm does not show any advantage.

Table 4.6: Average CPU time in seconds - population algorithm versus CPLEX - robust deviation

| population algorithm - robust deviation | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 1.154 | 0.909 | 0.760 |
| 60000 vertices | 2.557 | 1.918 | 1.648 |
| 90000 vertices | 3.864 | 2.895 | 2.469 |
| **CPLEX - robust deviation** | | | |
| | 5 children | 10 children | 15 children |
| 30000 vertices | 163.743 | 119.776 | 109.281 |
| 60000 vertices | 851.177 | 137.453 | 127.226 |
| 90000 vertices | 3294.592 | 188.199 | 142.982 |

Table 4.7: Average CPU time is seconds - population algorithm versus CPLEX - relative robust deviation.

| population algorithm - relative robust deviation | | | |
|---|---|---|---|
| | 5 children | 10 children | 15 children |
| 30000 vertices | 1.681 | 1.167 | 0.986 |
| 60000 vertices | 3.467 | 2.496 | 2.101 |
| 90000 vertices | 5.205 | 3.809 | 3.187 |
| **CPLEX - relative robust deviation** | | | |
| | 5 children | 10 children | 15 children |
| 30000 vertices | 250.002 | 104.695 | 80.759 |
| 60000 vertices | 1426.761 | 150.37 | 106.375 |
| 90000 vertices | 2217.743 | 199.58 | 142.479 |

# CONCLUSION

In this work we have explored robust variants of the maximum weighted independent set (MWIS) problem. Although the considered problem variants are NP hard, we wanted to find efficient algorithms for their solution. We were motivated by important real-world applications such as resource allocation.

First we have examined relationships between robust variants of our problem and the minimum weighted vertex cover (MWVC) problem. It is well known that the complement of a solution for the conventional MWIS problem is a solution for the conventional MWVC problem and vice versa. If such claim were true for robust variants, then algorithms for solving one problem could also be used for the other problem.

In Chapter 2 we have shown that the answer to the question "Is the complement of a robustly optimal independent set a robustly optimal vertex cover, and vice-versa?" is not straightforward. It depends on the chosen robustness criterion. Indeed, the answer is positive if the robust deviation criterion is used, and also if the whole collection of efficient solutions is considered. For absolute robustness or relative robust deviation the answer is in general negative, although some special cases exist when it is still positive. The results are interesting because they emphasize some important differences between conventional and robust variants of the considered optimization problems. They clearly show that our intuition and assumptions about problem relationships, acquired through conventional optimization, cannot be taken for granted when dealing with robust optimization.

Next, in Chapter 3 we have designed, implemented and tested an approximate algorithm for solving three robust variants of the MWIS problem corresponding to three robustness criteria: absolute robustness and (relative) robust deviation, respectively. The algorithm is based on evolutionary computing, and it relies on custom-designed crossover and mutation operators. We have proposed four crossovers, called AVX, MAVX, RVX and MRVX, together with five mutations, denoted by SRM, WIRM, WDRM, LSRM and CM. For testing, we have generated two test groups of robust MWIS problem instances. They are large enough to be nontrivial, but still small enough to be solved exactly by a general-purpose optimization package such as CPLEX. The first group contains denser graphs with 300 vertices, 30000 edges and 10 scenarios for vertex weights. The second group consists of sparser graphs with the same number of vertices and scenarios but only 20000 edges. Our algorithm turns out to be between 25 and 690 times faster than the exact algorithm provided by CPLEX. The actual speedup depends on the chosen robustness criterion and on the involved graph density. The solutions obtained by our algorithm are suboptimal, but their relative errors vs. the corresponding exact optima (measured in terms of robust objective-function values) are quite acceptable. With most efficient combinations of evolutionary operators, the expected errors range between 0.20% and 8%. Again, the actual percentage depends on the robust criterion and graph density. The operators within our algorithm work well because they successfully combine deterministic greedy techniques for constructing solutions with random processes that maintain the diversity of solutions. The results of Chapter 3 are interesting since they provide a practical way of solving large instances of the robust MWIS problem.

Finally, we have explored robust variants of the MWIS problem on trees. It is well known that the corresponding conventional variant is solvable in polynomial time. Moreover, there is an algorithm for solving the conventional variant with linear complexity in terms of number of vertices. In Chapter 4 we have proved that almost all robust variants of the MWIS problem on trees are unfortunately NP hard, which is a strong indication that such variants probably cannot be solved by a polynomial algorithm. Still, we can concentrate on designing specialized approximate algorithms which would take into account the involved special graph structure, rather than using the more general evolutionary algorithm from Chapter 3. Indeed, we have designed, implemented and tested such special algorithm. It combines elements from dynamic programming, evolutionary computing and greedy decision making. For testing, we have generated nine test groups of robust MWIS problem instances. We have test groups whose trees have 30000, 6000 and 90000 vertices, and for each number of vertices the maximum number of children has been set to to 5, 10 or 15, respectively. The algorithm is between 45 and 850 time faster than the exact algorithm provided by CPLEX. Again, the actual speedup depends on the chosen robust criterion and on the involved tree depth. The expected errors range between 0.08% and 8%. Again, the actual percentage depends on the robust criterion and tree depth. Although the errors are similar as for the evolutionary algorithm from Chapter 3, the specialized algorithm is much faster then the EA because it does not compute complicated crossover and mutation operators. Indeed, if we compare the size of test groups from Chapter 3 and 4, we see that in Chapter 4 we use noticeably larger graphs.

Putting it all together, although robust versions of the MWIS problem problem are NP hard, there still exist efficient approximate algorithms which can be used for their solution. We believe that our approximate algorithms are accurate enough for practical purposes and that their time requirements are tolerable.

# BIBLIOGRAPHY

[1] Aissi H, Bazgan C, Vanderpooten D: *Min-max and min-max regret versions of combinatorial optimization problems: A survey*, European Journal of Operational Research 197:427-438, 2009. ↑ 4, 5, 7, 14, 20.

[2] Aissi H, Bazgan C, Vanderpooten D: *General approximation schemes for min-max (regret) versions of some (pseudo-)polynomial problems*, Discrete Optimizaton 7:136-148, 2010. ↑ 1, 4.

[3] Ben-Tal A, El Ghaoui L, Nemirovski A: *Robust Optimization*, Princeton University Press, Princeton, 2009. ↑ 4.

[4] Bertsimas D, Sim M: *Robust discrete optimization and network flows*, Mathematical Programming 98:49-71, 2003. ↑ 4.

[5] Bertsimas D, Sim M: *The price of robustness*, Operations Research 52:35-53, 2004. ↑ 4.

[6] Bertsimas D, Brown DB, Caramanis C: *Theory and applications of robust optimization*, SIAM Review 53:464-501, 2011. ↑ 4.

[7] Brandstädt A, Lozin VV, Mosca R: *Independent sets of maximum weight in apple-free graphs*, SIAM Journal of Discrete Mathematics 24:239-254, 2010. ↑ 20.

[8] Chen GH, Kuo MT, Sheu JP: *An optimal time algorithm for finding a maximum weight independet set in tree*, BIT 28:253s-256, 1988. ↑ 3, 60, 63, 69.

[9] Ehrgott M: *Multicritera Optimization*, 2nd Edition, Springer, Berlin, 2010. ↑ 7, 31.

[10] Eiben AE, Smith JE: *Introduction to Evolutionary Computing*, 2nd Edition, Natural Computing Series, Springer, Berlin, 2015. ↑ 37, 38, 39.

[11] Frank A: *Some polynomial algorithms for certain graphs and hypergraphs*, In: *proceedings of the 5th British combinatorial conference*, 211-71, University of Aberden, 1975. ↑ 60.

[12] Garey MR, Johnson DS: *Computers and Intractability: A Guide to the Theory of NP-Completness*, W.H. Freeman, San Francisco CA, 1979. ↑ 2, 60, 63, 67.

[13] Gross JL, Yellen J, Zhang P (editors): *Handbook of Graph Theory*, 2nd Edition, CRC Press, Boca Raton FL, 2014. ↑ 8.

[14] IBM ILOG CPLEX Optimization Studio: *CPLEX User's Manual, Version 12*, Release 8, IBM Corporation, 2017, https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0. Accessed 10 October 2018 ↑ 23, 49, 51.

[15] Jungnickel D: *Graphs, Networks and Algorithms*, 4th Sdition, Springer, Berlin, 2013. ↑ 8.

[16] Kasperski A, Zielinski P: *Complexity of the robust weighted independent set problems on interval graphs*, Optimization Letters 9:427-436, 2015. ↑ 16, 20, 23, 60, 61.

[17] Kasperski A, Zielinski P: *Robust discrete optimization under discrete and interval uncertainty: A survey*, In: Doumpos M., Zopounidis C., Grigoroudis E. (editors), *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, Springer, Cham CH, 2016. ↑ 6, 7.

[18] Korte B, Vygen J: *Combinatorial Optimization - Theory and Algorithms*, 5th Edition, Springer, Berlin, 2012. ↑ 10.

[19] Kouvelis P, Yu G, *Robust Discrete Optimization and its Applications*, Springer, Berlin, 2010. ↑ 1, 4, 5.

[20] Lobato FS, Steffen V Jr: *Multi-Objective Optimization Problems: Concepts and Self-Adaptive Parameters with Mathematical and Engineering Applications*, Springer, Berlin, 2017. ↑ 31.

[21] Mandal S, Pal M: *Maximum weight independet set of circular-arc graph and its application*, J. Appl. Math. Comput. 22:161-174, 2006. ↑ 60.

[22] Michalewicz Z: *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Edution, Springer, New York, 1996. ↑ 37.

[23] Microsoft Corporation: *Visual Studio Documentation 2017*, https://docs.microsoft.com/en-us/visualstudio/?view=vs-2017 (2018). Accessed 10 October 2018 ↑ 51.

[24] Oracle Corporation: *Java Documentation*, https://docs.oracle.com/en/java/. Accessed 22 August 2019. ↑ 78.

[25] Pal M, Bhattacharjee GP: *A sequential algorithm for finding a maximum weight k-independent set on interval graphs*, Int. J. Comput. Math. 60:205-214, 1996. ↑ 60.

[26] Puljić K, Manger R: *Comparison of eight evolutonary crossover operators for the vehicle routing problem*, Mathematical Communications 18:359-375, 2013. ↑ 38.

[27] Saha A, Pal M, Pal TK: *Selection of programme slots of television channels for giving advertisement: a graph theoretic approach*, Info. Sci. 177:2480-2492, 2007. ↑ 60.

[28] Sakai S, Togasaki M, Yamazaki K: *A note on greedy algorithms for the maximum weighted independent set problem*, Discrete Applied Mathematics 126:313-322, 2013. ↑ 39.

[29] Talbi EG: *Metaheuristics - From Design to Implementation*, Wiley, Hoboken, 2009. ↑ 37, 38, 39.

[30] Talla Nobibon F, Leus R: *Robust maximum weighted independent set problems on interval graphs*, Optiization Letters 8:227-235, 2014. ↑ 13, 14, 16, 20, 22, 60, 61.

# APPENDICES (ON CD)

# BIOGRAPHY

Ana Klobučar was born in Osijek, Croatia on 27 May 1991. She graduated from III. grammar school in Osijek in May 2010, and from the University of Zagreb, Faculty of Science, Department of Mathematics in July 2015. summa cum laude. She is currently working as a teaching assistant od mathematics at the University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture. Also, she is a member of the Croatian Science Foundation project ”Efficient algorithms for robust discrete optimization”. Besides mathematics, her other interests are dancing, travelling and painting.

# ŽIVOTOPIS

Ana Klobučar je rođena u Osijeku, Hrvatska 27. svibnja 1991. Završila je III. gimnaziju u Osijeku u svibnju 2010. Diplomirala je na Sveučilištu u Zagrebu, Prirodoslovno matematički fakultet, Odjel za matematiku u srpnju 2015. sa *summa cum laude*. Trenutačno je zaposlena na Sveučilištu u Zagrebu, Fakultet strojarstva i brodogradnje kao asistent matematike. Također, član je projekta Hrvatske zaklade za znanost pod nazivom „Efikasni algoritmi za robusnu diskretnu optimizaciju“. Osim matematike, drugi interesi su joj plesanje, putovanje i slikanje.